Health Insurance

Evaluation System

*Task02 UPMC Report*



**Team 9**

Team Member: Misty Yang, Josie Liu, Shuoze Wang,

Yanlin Liang，Jiaqi Wu 

## 

## 

## Ⅰ. Executive Summary

The ebConsultants LLP provides technical solutions in various areas including Healthcare, Banking, Retail and Logistics. Our team has five consultants, each specialized in a particular area, including technology, operations, management, and so on. With the 5 experienced core staff and a passionate team, we are able to provide creative and valuable solutions for UPMC.

Our client, UPMC,has more than 25 hospitals, 6000 licensed beds, nearly 70,000 employees, and was ranked No. 12 in the Nation in 2016. UPMC wants to attract more healthy people to enroll their health insurance plan by offering reduced healthcare rates to people with healthy lifestyle. However, the traditional way to assess people’s lifestyle is unreliable and time consuming.

Our system can provide UPMC a more accurate way to evaluate client's’ lifestyle using the technology of cloud computing. The wearable camera on clients will capture and upload images related to lifestyle, and upload to the cloud. The cloud will run a function to generate a health score by analyzing these images and return to the user.

We believe the two most significant factors in a person’s lifestyle would be diet and exercise. Thus, our system focuses on processing images related to **diet, alcohol and exercises.** Also, we take **drugs** as a significant factor when conducting the evaluation. Once a person is determined as a drug addict by our system, there would be no chance for him/her to be evaluated as healthy.

Besides, this system is highly scalable. One web applications can support multiple users. The web applications can be deployed on Docker containers and these Docker containers can be deployed automatically by Dockerfiles. In addition each virtual machine for dockers can be launched by auto horizontal scaling if there are a large number of requests. Also our API (AWS API Gateway) triggers of AWS Lambda functions for calculating scores can be integrated to other mobile and web applications of UPMC as micro-services.

With this revolutionary system based on cloud computing, UPMC will be able to launch a brand new health insurance program. The assessment of health level provided by our system will give the possibility for UPMC to attract more people who lead healthy lives and encourage less unhealthy clients.

## Ⅱ. Problem

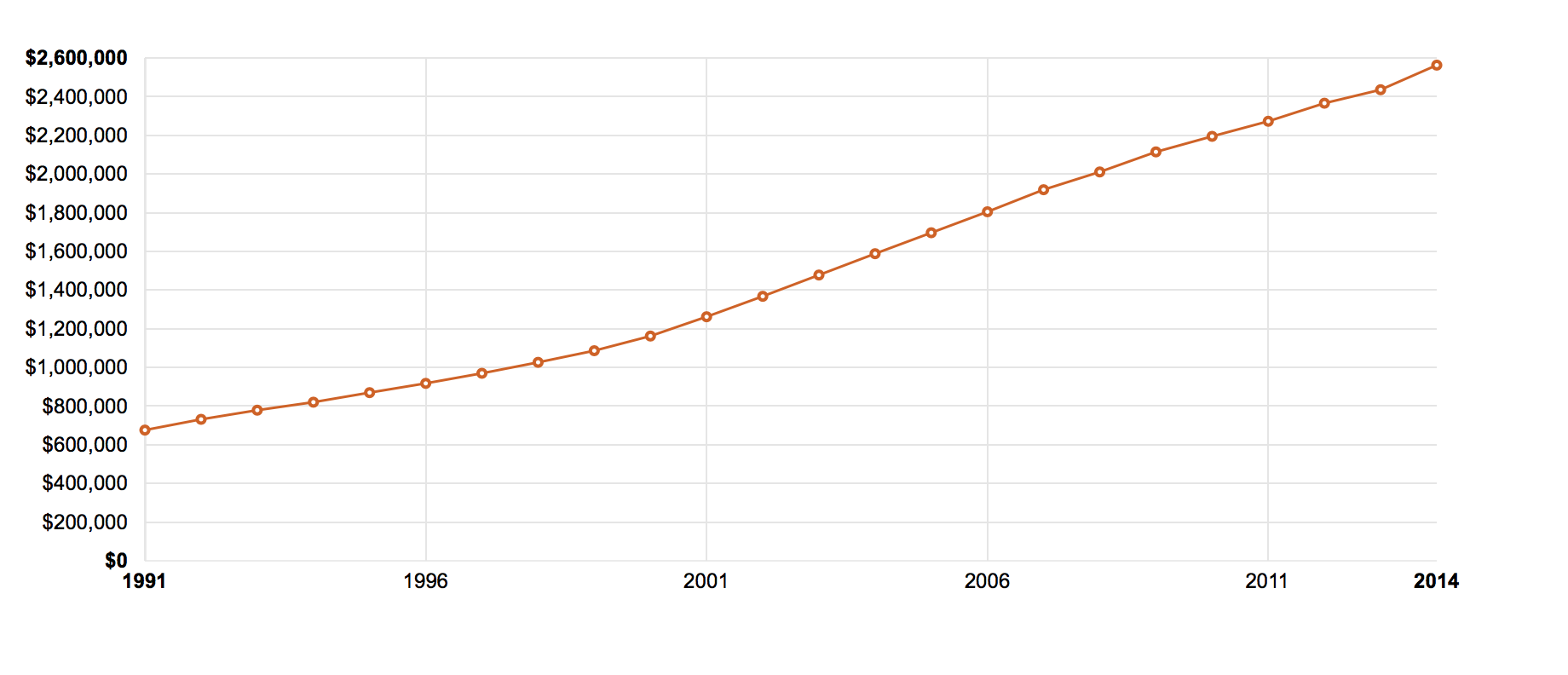


Fig. Health Care Expenditures by State of Residence(CMS,2017)

Nowadays, the price of medical insurance in America goes higher quickly, which gives more and more people huge economic pressure. Under this kind of circumstance, UPMC is going to give a discount on medical insurance to those people who have more healthier life habits. And with implementing this new insurance plan, UPMC will attract more people with healthy life habits to choose UPMC. In addition to this, the plan of UPMC would stimulate more people to adopt a healthy lifestyle.

As we all know, the new insurance plan is fantastic. People who choose UPMC can have a more healthy lifestyle and save money on medical insurance. However, the most critical thing is how to measure or evaluate the behaviors of people who want to participate this plan. And UPMC has huge difficulties on how to create practical criteria to determine whether the lifestyle of one person is healthy and what is the healthy degree of their lifestyle.

Actually, it is a cliche to reduce the cost of healthcare for healthy people. But the most critical question is that we can not easily decide one person is healthy or not. And even if someone is healthy, we can not figure out how healthy this person is by giving some direct comparable values. In recent years, many insurance companies would like to provide this special discount healthcare service. They determined one people by submitting health forms, body mass index or even by their disease history. Although these companies take time and money to investigate the health conditions of one client, they always get inaccuracy data in the end.

By health forms filled by clients, companies can not get what happened actually because many people like to lie in this kind of forms. And by body mass index, there are always many people who are unhealthy but with a normal body mass index. So it is not feasible indeed. For disease history, it is not rational to decide clients’ present health condition by analyzing their disease history. All of these limitations should be solved by new advanced technology scientifically.

Under this kind of circumstance, our Smart UPMC insurance evaluate system would like to take responsibility to create a feasible and practical method to give a rational discount for healthcare. Our team will provide a lightweight low-power wearable camera for the client as a sensing device to get the diet and exercise information. And this device can capture the sensing pictures and upload them to cloud. These data would trigger the event and our function on cloud service can analyze these data to know the diet and exercise information.Then we can evaluate this info for each client so that we can decide if we would offer them a discount insurance policy. We choose the pictures when the clients are eating or exercising for about one month or longer. So we can evaluate the pictures by cloud computing function to detect the label quickly and automatically. And we can process pictures of very huge amount of people just by cloud service. What we need to do is to design the whole process carefully and make a perfect system for UPMC.

## Ⅲ. System Design and Implementation

### 1.Overview

We already know that some companies try to have discount health insurance plans with some limited methods. And there is no way to attract more clients by these ways because many of them are time-consuming and inaccurate. So we decide to put lightweight low-power camera for client to capture information. And this method is convenient and scientific which would not invade the privacy of clients and would not disturb their daily life because the camera is so small. And after this step, what we need to do is to get enough and accurate information from the pictures the camera shoots. And the camera can reuse for many times to reduce cost.

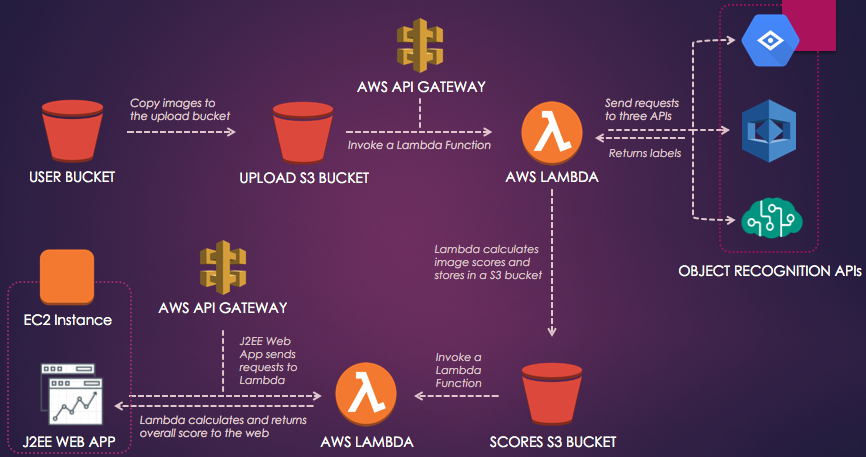
Our functions of system are cater to the requirements from clients. First, we can not get any other information other than images. So our task is to analyze these pictures carefully to get exactly what we need. We team decide to choose images from their food, drinks, drugs and exercise. We use all of these aspects to evaluate their health habits with score, and then we can set a score line to filter the clients we want.

The fact is that if we have one client to evaluate, we could analyze almost 100-200 photos in one month period. But UPMC has millions of insurance clients to evaluate which traditional process method can not deal with. That is because the pictures of millions of people would take huge hardware memory device to store. Besides to storage problem, we also need high-speed processing unit to analyze all the photos quickly. Based on these conditions, it is impossible for UPMC to investment lots of money on hardware and CPU to figure out one insurance problem. As a result, our team is going to design our system based on cloud computing service, which we can not only store all the image in it but also call functions to process pictures by cloud computing processing unit. What we only have to pay is the service fee, which is much cheaper than buying all of the storage and processing facilities.

### 2. Design Structure

And we found there are many cloud computing service providers. What we should do firstly is to design the whole structure by our demands. Then we can decide which service provider is the best choice to accomplish our requirements.

Firstly, for each person we have hundreds of pictures about diet and exercise and so on. We should figure out that our team do not have too many storage to save all the data. In order to make everything clear, we apply Amazon Web Service as basis to show our system architecture. As the chart shown below, user S3 bucket should firstly transfer all the data to upload S3 bucket. After this step, the AWS LAMBDA will take pictures from this bucket by permission of credential. And we have already coded some programs in LAMBDA. LAMBDA can process all of the pictures follow our instructions.



It would call 3 different APIs to detect all the labels in pictures. Why we choose 3 APIs to be involved is that we think it can reduce risk of failure of recognition when picture is in extreme situation. For example, the rest APIs can work normally when one of the vision recognition API can not figure what the picture contains. Now, we can use multiple threads to request labels for corresponding pictures. To avoid possible misleading information because of difference of response time, we are going to give every image an index. This index is not only used in uploading procedure but also used in giving tags to returned results from APIs. And when we want to show labels of each picture, we could track the result by index value.

All the returned labels and index would be stored in another S3 bucket---SCORES S3 Bucket. Another LAMBDA can upload the data in scores bucket to calculate the total score and we focus on the score. Each picture should has a score to display its effect on the final insurance discount decision. In order to consider every tiny labels into the score, we create a dictionary to store vocabularies related to health conditions. For each element in the dictionary, we set a pair value to indicate the attribute of one label. If one label is benefit to one’s health, it will be a positive value, vise versa. Beside to this, we also give weight value to different API because some API performs better than other API. So we calculate each point of the label and multiply with the weight value. And then we add all the label points to get the total score of one picture. And we can add all of the scores of pictures to get a final one.

As shown in the chart, UPMC insurance manager can log in the web app by credentials and request health information for certain client. The manager input the id of client and then web can returned one scatter diagram. Each scatter point represent one picture with index and score. And the total score for the client would also shown in the webpage. So the manager can judge whether give this client the discount right.

And we can see, there is no need for UPMC to purchase hardware and software, All UPMC need to do is to hire people like us to develop the whole system and pay for the cloud server service fee. By this way, UPMC can reduce a lot of cost. And there are more advantages:

(1) The whole evaluation process is transparent to both client and UPMC manager. What they just to do is keep their routine life and wait for the result without doing anything.

(2) By using cloud service, it is more accurate and safe. Because now nobody can lie in the health habits, every results are calculated by advanced recognition API and algorithm scientifically. And nobody can access these data without credentials, which improved the privacy safety.

### 3. Algorithm

In this section, we would focus on introducing details of our algorithm. So we already know we would get labels of pictures from LAMBDA function. Now the question is: how to give score to evaluate certain labels? In other words, how our algorithm operates the final total score?

Firstly, we have run around 5,000 pictures to collect enough labels about food, exercise and drugs. After this step, we divided these labels into 3 sets and store them into dictionary. One is positive set and words in this set could benefit clients’ health. And obviously other two sets are neutral and negative. So we can assign positive value to words in positive set. We set no value for neutral set and assign negative value to negative set. Owing to that we can not make clear the specific amount of the each food material, we give the equal score to all the healthy food like fruits and vegetables. And we assign negative value to all of the unhealthy food such as junk food and fast food. And we assign little part of food labels as neutral value because these food do not have apparent effects on body health. As it is very important to assign correct value to different labels, we took a lot time researching on value dictionary. Our team gives each label a proper score to make the whole algorithm logically and scientifically.

Though we can very subtle and accurate score on each label. We team tried hard to find other ways to improve our accuracy. Here are some other methods we used:

① Using all of the APIs

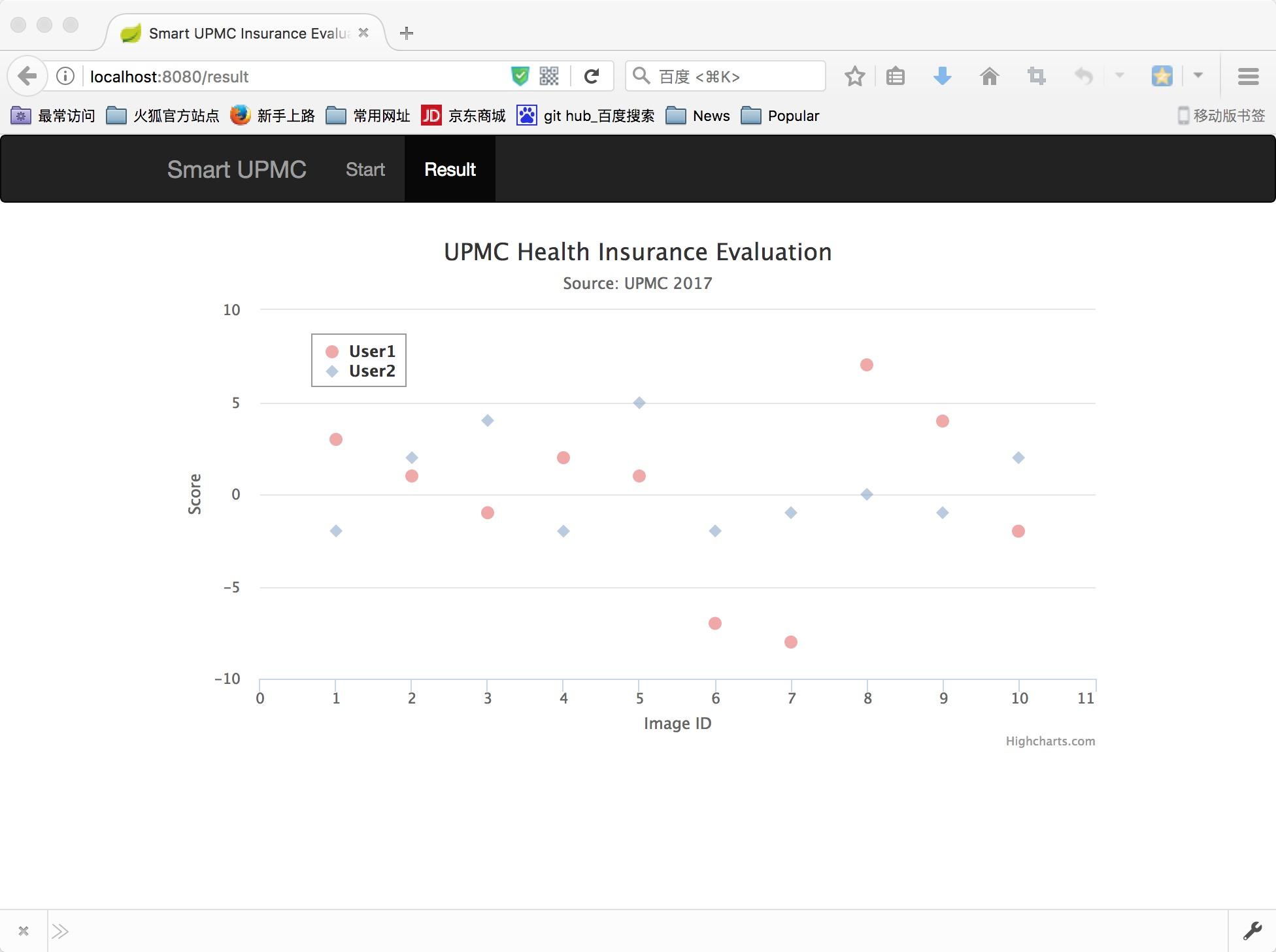
As we all know, there are 3 main cloud vision recognition service providers. Though they have different quality of performance, we still want to use all of the APIs to calculate the total scores. Why? Because using multiple APIs can reduce the risk of system crash when one API has difficulty in detecting the labels. Of course, only using 3 APIs would damage the effect of algorithm. But we introduce the weight value into different APIs score. Based on analysing responses from different APIs testing the same picture bucket. We measured their accuracy by analyzing the number they detect correctly. So we make weight value to different API by their performance. And the weight value would be multiplied into the total score.

②Considering time

We take the time factor as consideration in calculating the total score. We think people is always changing. So we focus more on their recent information about diet and sport. Based on this, we create a dynamic linear coefficient to represent the time factor. This coefficient could increase with time. So the more recent pictures could receive more weight in scoring.

③Multiple Users

We can support multiple users to show and calculate their scores in the same screen. We save all the data for users in one bucket and we can call the information to one UI.



## Ⅳ. Testing and Experimentation

Our smart UPMC insurance evaluate system has experienced many tests to make it stable and reliable.

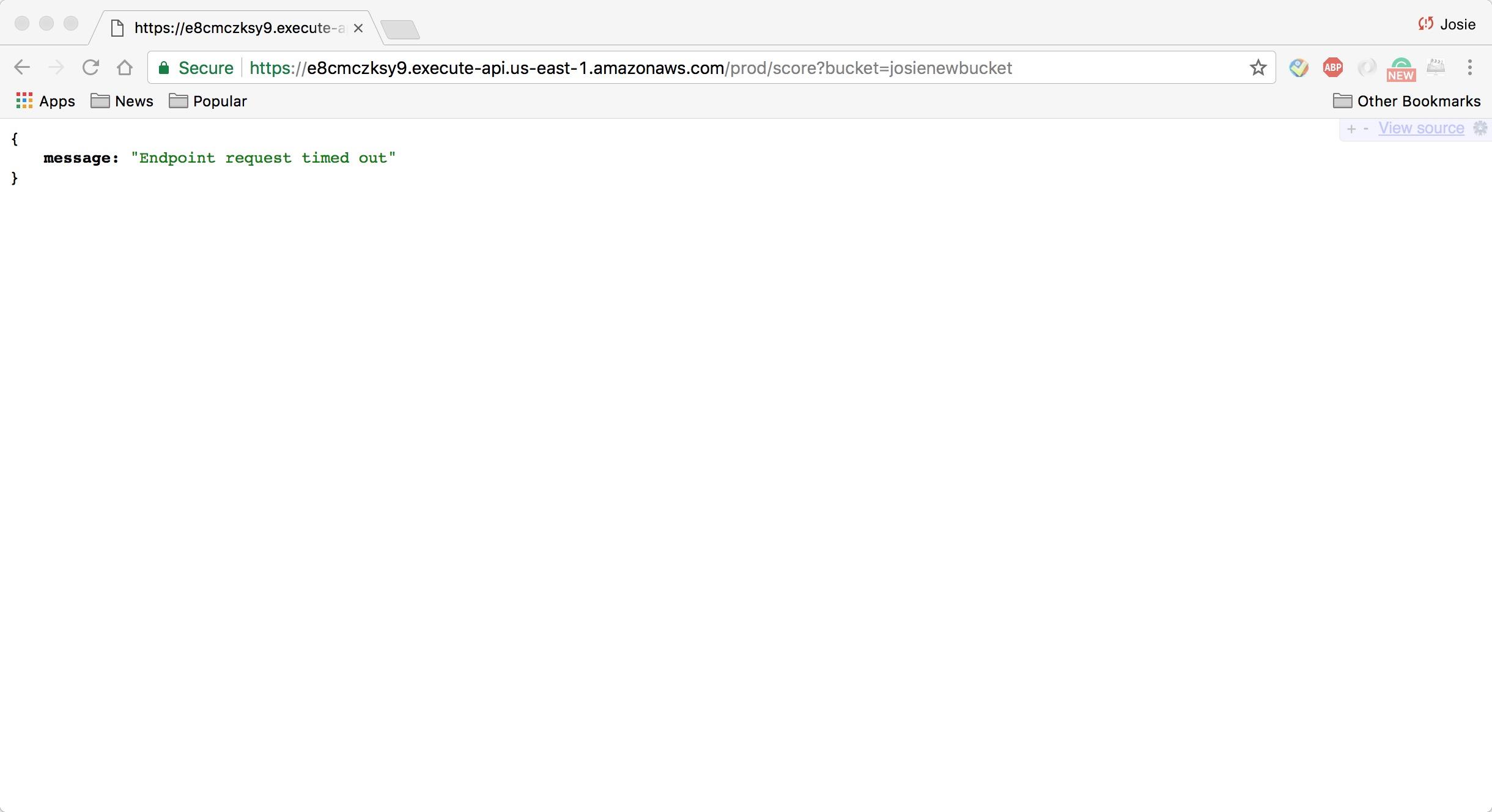
* Speed Test

Speed is the main factor of system performance when we built our system. We have struggled several time with long processing time during the developing procedure.

* Multi-thread to call APIs

We test our system with the method of calling 3 APIs one by one at first, and the average processing time for one pic is over one second. Then we use multi-thread to call 3 APIs separately, the processing time is down to about 500ms per image. Thus, we decide to use 3 threads to make API calls.

* Multi-thread to process large amount images



Above is the output when we try to test 100 images. The end point time out error is because time limitation when calling a certain APIs to connect one cloud computing server. The problem occurred very frequently especially when using Azure Computer Vision API.

We test our system by using multi-thread when processing different images. After this test, we decided to apply multiple threads in picture processing, and the time out error problem never happen again. This only apply to the condition in which we are trying to process lots of images in the same time.

* Accuracy

To improve our accuracy, we also conduct many experiments to adjust different weight factors of the three APIs.

* Accuracy with same weight factors

When considering three APIs without any differences, the output was not accurate enough. It returned “0” several times , which represents “non-food” when we uploaded food images. Our system couldn’t treat image with a Kitkat chocolate bar on it as a food image at first. After conduct tests on three APIs thoroughly, we figured the reason was the way we use to decide if the image is a food images doesn’t work well with the Azure API. So we decided to give lower weight to when calculating image score.

* Accuracy with different weight factors

When considering the health classification, we tested our system with both healthy and unhealthy food. During these tests, we tried to figure out the best weight factors to give to improve our accuracy.

After analyzing the characteristics of each API, we found that Amazon Rekoginition is usually good at distinguish food and non food, because it always return the “food” label with a high confidence score when we test a food image. However, recognize different food is not its speciality. It usually returns high-level labels like “asian food” while the Google Vision API usually returns more specific labels, which is not enough to determine if the food in that image is healthy.

After testing different pair of weight factors, we found the best pair would be {Amazon = 0.1, Google = 0.8, Azure = 0.1}.

## Ⅴ. System Evaluation

### Evaluation.

Our system has advantages in at least 4 aspects.

* Scalability

Our system is easy to scale. If millions of users use our system, we can use horizontal scaling to add more servers for our web applications within minutes by Dockerfile and docker images. We are going to Dockerfile to create Docker images and run docker containers within one instance.

If we have to process more data, we will take into account the auto-scaling technique we learned previously. Then, we will use EC2 instances instead of lambda function, and we can adjust the number of EC2 instances according to the number of pictures we need to process. Now we only need to process 10 pictures for 20 seconds, in the future we can process more than 100,000 pictures in 1 minute. By that time, this system will be really commercial available. The insurance company could use this system to process the information of thousands of patients almost instantly. Our system has a very smart mechanism to scale the ec2 instance automatically.

Besides we are using AWS API gateway to trigger lambda functions and the insurance evaluation services can also be integrated with mobile app and other web applications.

We are also going to launch a load balancer to disperse flow in our system during the peak-hour if the customer population grows to million-level.

In addition since we are using cloud services, we can easy to scale to different regions within a few minutes by launching instances in different areas.

To deal with speed problems during scaling, we are going to use DynamoDB as database instead of storing image scores in a S3 bucket if millions of users are using our systems. DynamoDB is also a service provided by AWS, which makes our system more scalable.

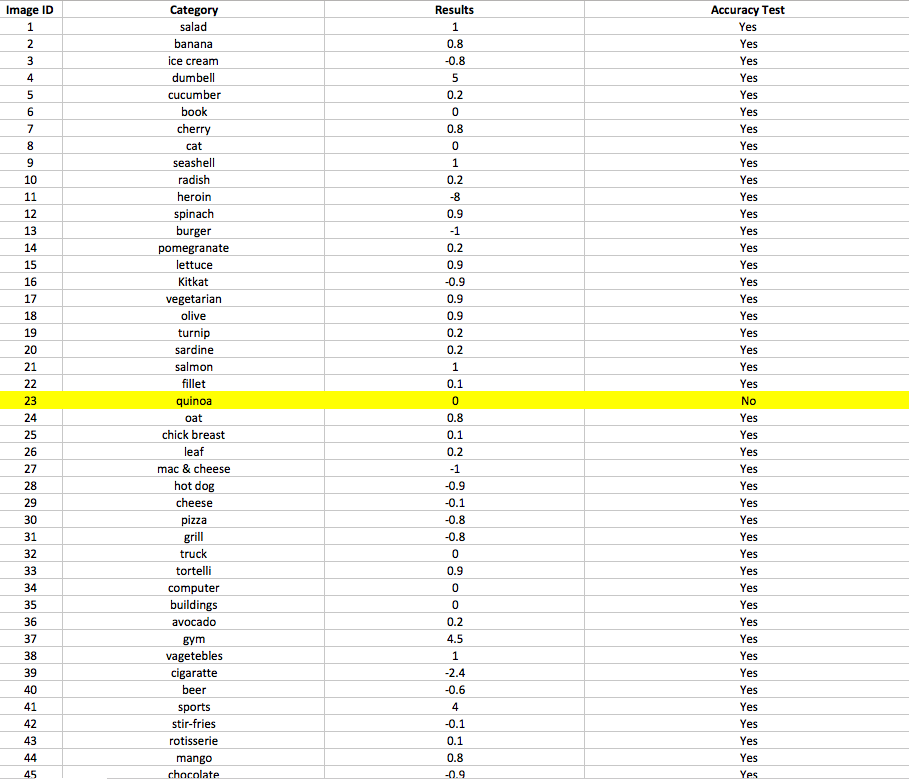
* Accuracy

To make the system more accurate, we use **all three object recognition APIs** instead of just using one.

Based on our tests, our system can reach 97% accuracy (test table see below).

And to improve accuracy we gave **different weighting factor** to 3 APIs. One user image have a total score range [-1, 1]. Google Vision API weights 0.8 points because it is the most accurate after numerous testing and the other two APIs only counts 0.1 each. So for example, if a picture is recognized as healthy food by all three apis, then it has 1 points. If only Google api returns as healthy, then it has 0.8 points.

To improve the accuracy of assessing healthy habits, we also take **exercises, drugs, cigarette and alcohol** into consideration, these three will all mark down the health scores in different degree.



* Performance

By implementing multithreading in lambda function, our system can process one pic in less than 1 second (about 500 ms).

* Cost

Since UPMC needs a reasonable price to build such a system, we use cloud computing service so that UPMC doesn’t need to make investment in building and managing data centers and all hardware devices.

It will cost $244K to launch our system with one million users, about 0.2 dollars per customers. To scale by 1,000 users, it will cost 15K. To scale by 10,000 users, it will cost 18K.

### Limitation & overcome strategy

Due to the limitation of budget and time, our solution still has some limitations.

* Accuracy with complexity

Our solution might be lack of accuracy when it comes to complicated lunch boxes.

Our system can not recognize food quantity and can’t evaluate from the nutrition aspects.

* Single standard

For now, our system only simple treats high-calories food like pizza and burgers as “unhealthy” and healthy food like salad as “healthy”. However, diet evaluation does not have simple single standard. For instance, standards should be different between overweight and underweight clients, we should associate food quantity with client’s weight, etc.

* Speed

It’s possible to achieve shorter processing time. For now we have about 500ms processing time for one picture.

* UI

A more professional and multi-functional UI is possible if we have more time.

Solutions

* Adding dietary balance and calorie measure APIs
* Associating the image recognition analysis with client’s fitness records
* Training more data to expand our food dictionary to improve accuracy
* Spending more time on UI design to build more user-friendly interface

## Ⅵ. Improvement

Firstly, with the help of our system, UPMC health insurance will not need to worry about the scalability. Even if the UPMC health insurance can not predict the user number in the beginning, because of the scalability of computing clouding, the system can ensure the right quantity of capacity to handle current traffic. Additionally, because of the auto scaling of cloud computing, the system can automatically increase and decrease capacity as needed. So the UPMC health insurance can save a lot of investment cost and maintenance cost by launching exact number of virtual machines when the system actually need and terminate the virtual machine if they are unnecessary.

Secondly, by implementing our system, the customers of UPMC health insurance can directly see his scores of food in the past several months, which can totally encourage them to eat more healthy food and establish healthy lifestyle. Also, in this charts, users can clearly distinguish healthy or unhealthy food pictures from non-food pictures, the system will provide a more accurate and clear path for customers to live a healthy life and earn a discount in insurance. The system provide a more efficient and precise way for UPMC health insurance to evaluate the user’s health condition and offer veracious discount to attract users or retain the existing customers.This system helps people to stay healthier and live a more fulfilling live. Also, the rise of chronic disease conditions threatens people’s health. This food evaluation system help people to conquer bad dietary habits. According to a report conducted by the Go365 company, after implementing health rewards system like ours, numerous families and people benefits from such policy. Not only because they receive insurance discounts because of their healthy lifestyle, but also because they avoid suffer from chronic diseases causing by high- fructose or high- fat diet.

Thirdly, UPMC will benefit from this accurate evaluation system. Most importantly, nowadays although a lot of people live a healthy life, their cost of insurance continue to rise. By implementing this system, UPMC’s Healthy Living Insurance Program will be able to offer discounts or reduce health care rates to these people who make good lifestyle choices.

In this way, because economic factors is a significant factors to attract customers, UPMC’s Healthy Living Insurance Program will attract clients with healthy lifestyles. In a word, this system offers evaluation is like the telematic-based tracking but performs a lot more better.

## Ⅶ. Cost Estimation

### 1.Assumption

* Users number

To assess scalability, we calculate cost on three users tiers:

1,000 / 10,000 1,000,000 (1 million) users

* Time period

Assume the new health insurance program need 30 days to collect images from one user.

* Number of images

Assume there will be 1 image for 1 meal of 1 user. Then the total number of images on three user tiers would be 90,000 / 900,000 / 90,000,000

* Avg. size of images

Considering the size of original photos taken by cameras and phones, we think the average size of images is 1MB.

### 2.Services

In this cost analysis section, we calculate cost on AWS Services, including Lambda, S3 storage, S3 Request, API Gateway, and EC2; cost by calling Amazon Rekognition API, Google Vision API, and Azure Computer Vision API; labor cost including developing, deploying, maintaining, and training fee.

For system developing and deploying, since the AWS is able to conduct auto scaling, we will only consider developing once. It is $90,000 (see below). The rest of cost considering this goes to maintenance cost.

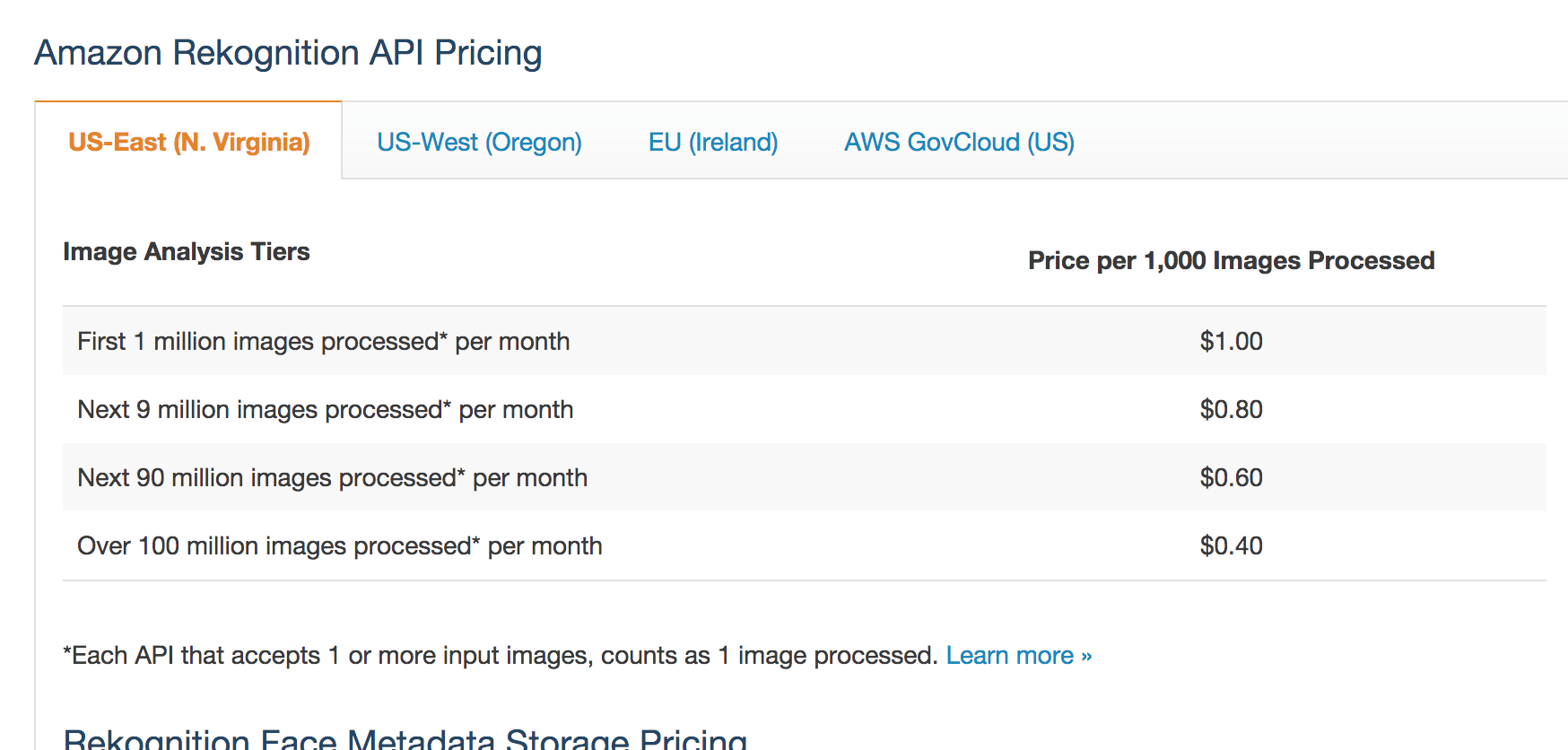
**Chart of Cost for one-month based on three tiers**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Category | Service | Price | | | Subtotal |
| API | Amazon Rekognition API | 1,000 users | 10,000 users | 1 million users | 1st tier: $313.5  2nd tier: $3,135  3ed tier: $210k |
| $90 | $900 | $58,562 |
| Google Vision API | $133.5 | $1,335 | $92,500 |
| Microsoft | $90 | $900 | $58,500 |
| Amazon Web Service | Lambda | $0 | $0.16 | $213 | 1st tier: $15k  2nd tier: $15k  3ed tier: $33k |
| S3 Storage | $1.1 | $11.26 | $1,125 |
| S3 Request | $0.88 | $8.8 | $880 |
| API Gateway | $3.5 | $3.5 | $7.0 |
| EC2 | $15,298 | $15,298 | $31,104 |
| Labor | Training Fee | $4.3 | $4.3 | $8.6 | $4.3; $4.3; $8.6 |
| Maintaining | $0 | $0 | $1,500 | 0; 0; $1,500 |
| Total | | $15k | $18k | $244k | - |

### 

### 3.Explanations

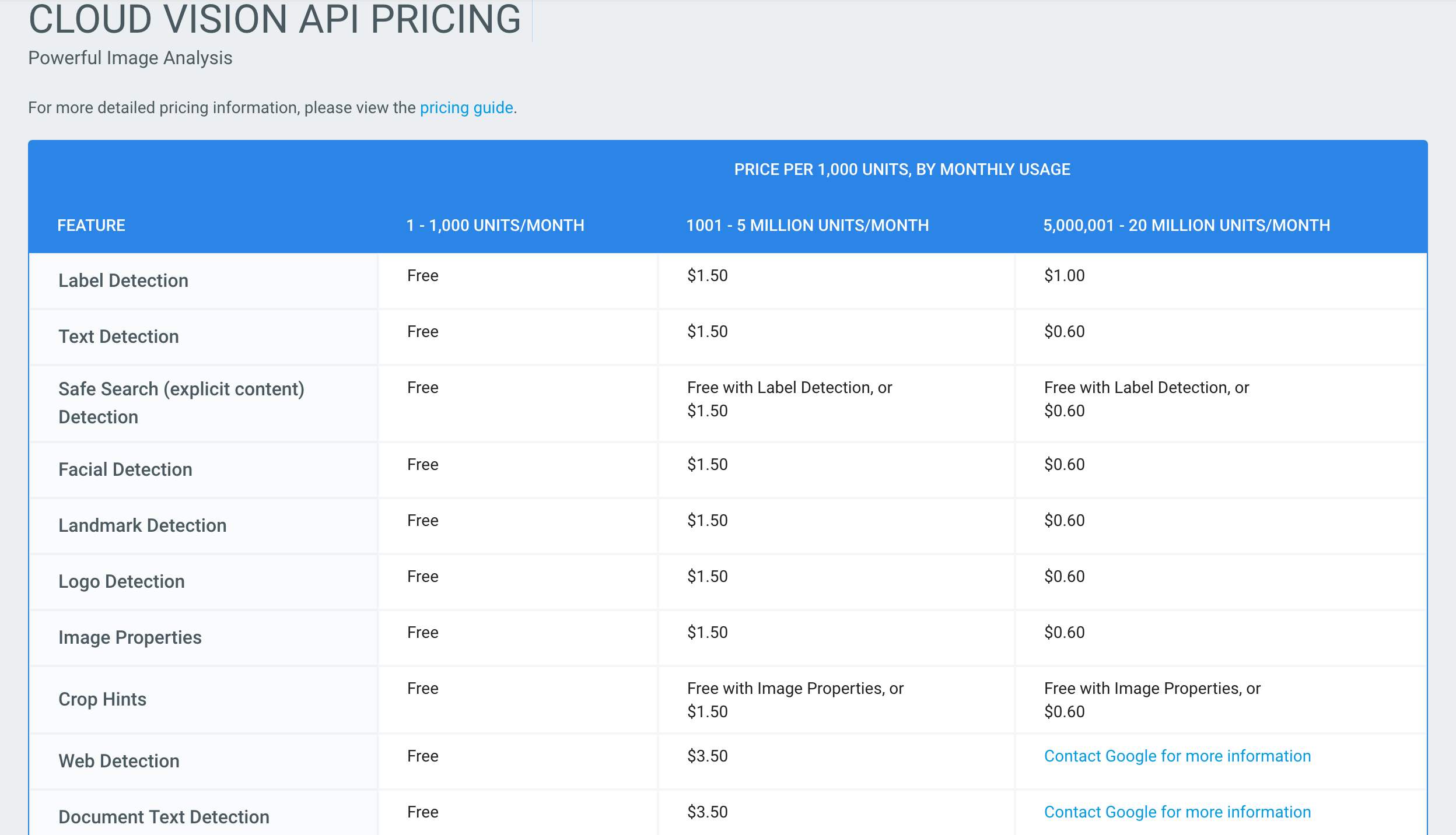
* Amazon Recognition API:



Number of images on three user tiers are 90,000 / 900,000 / 90,000,000. For tiers with less than 1 million images, our system will be using first pricing tier. For one million users, we use the ladder price.

Total cost: $90 / $900 / ($1 \* 1,000 + $0.8 \* 9,000 + $0.6 \* 80,000 = $56,200)

* Google Vision API:

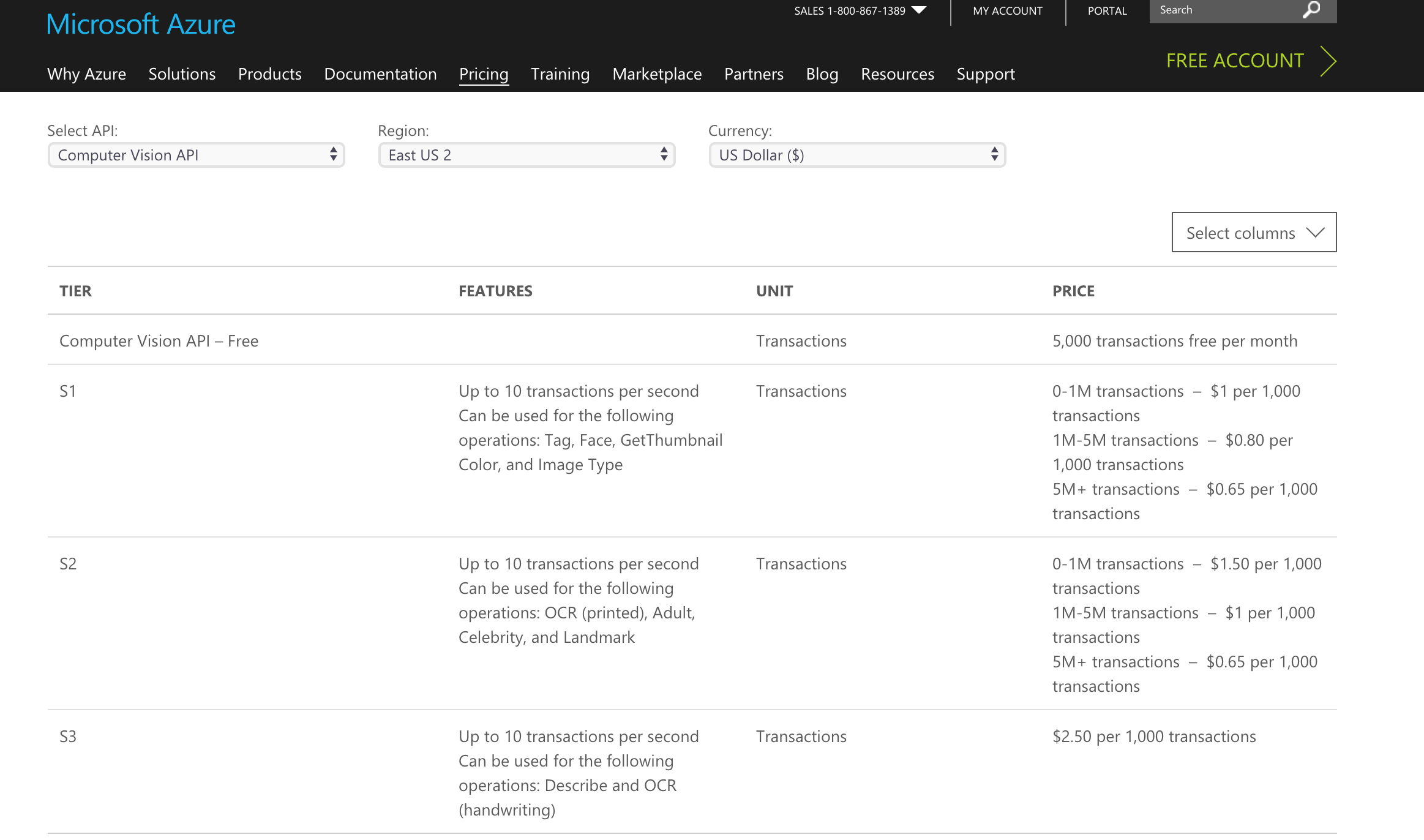


Our system will only use Label Detection Service for Google Vision API.

Number of images on three user tiers are 90,000 / 900,000 / 90,000,000 .

Cost: $133.5 / $1335 / ($85,000 + $7500 = $92,500)

* Azure API:

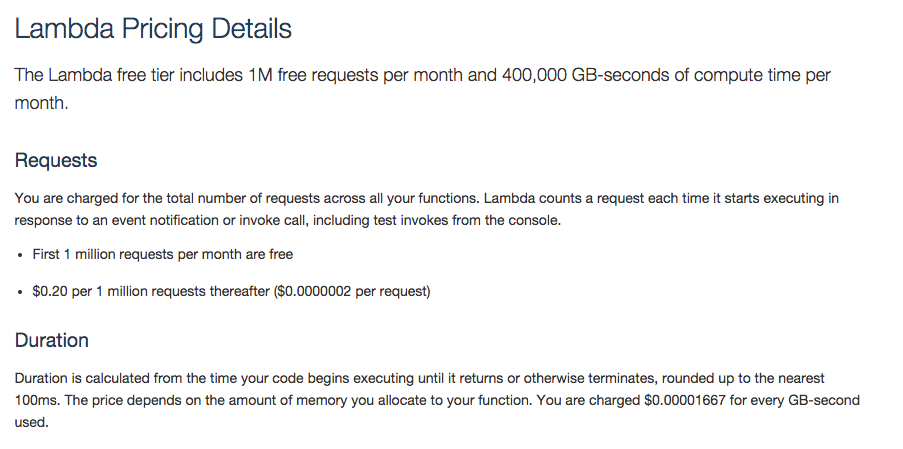


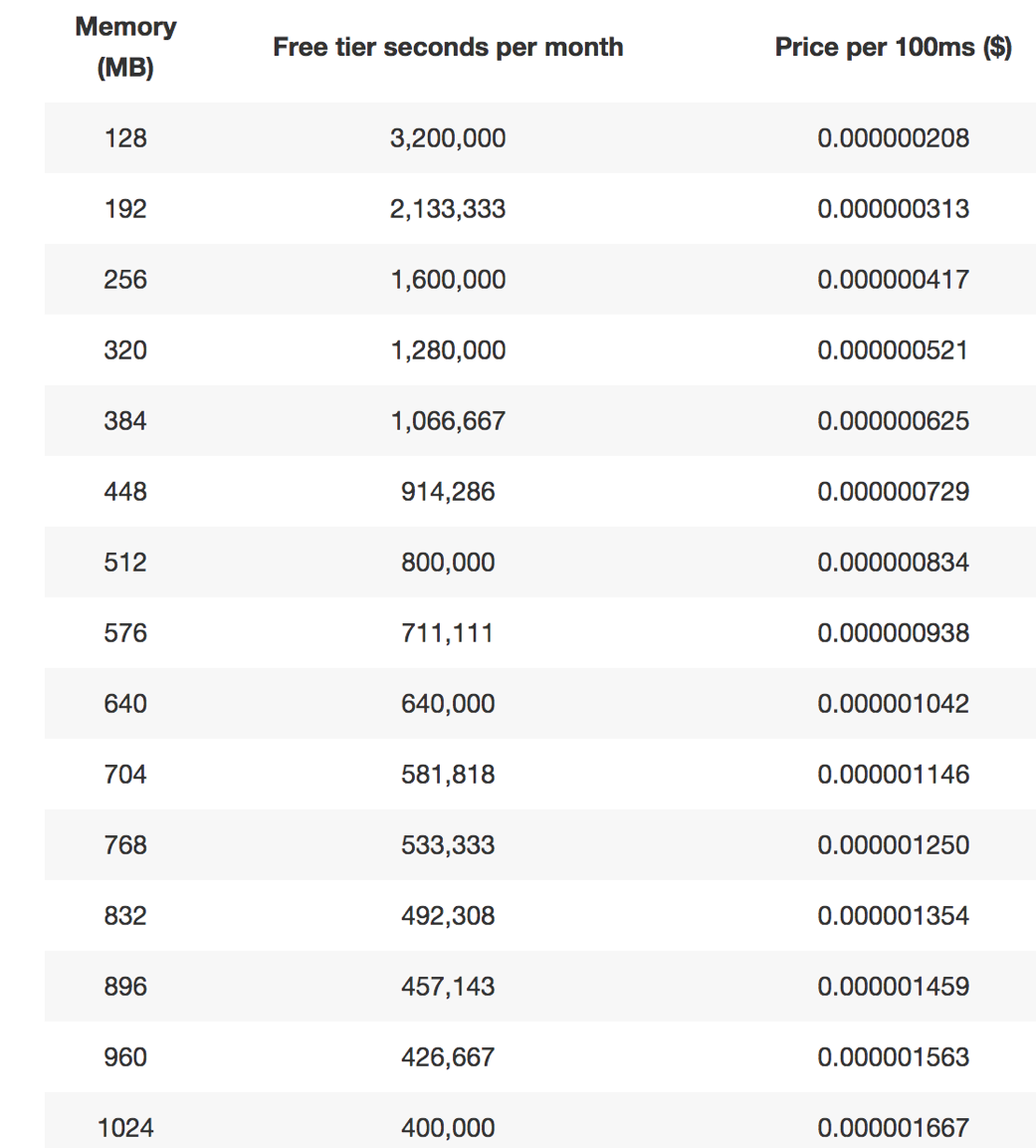
For tags detect service, we will be using S1 tier service of Azure Computer Vision API.

Number of images on three user tiers are 90,000 / 900,000 / 90,000,000.

Cost: $90 / $900 / $58,500

* Lambda Function





The total cost on the Lambda Function is compute charges plus request charges.

Our code is about 120MB. We need to allocate 128MB of memory to the lambda function. This has nothing to do with the user number. We have 2 Lambda for process each image, so we will execute 180,000 / 1,800,000 / 180,000,000 times in one month, and it ran for 500ms each time.

(a). Compute charges

The monthly compute price is $0.00001667 per GB-s and the free tier provides 400,000 GB-s.

Total compute = (180,000 / 1.8M/ 180M) \* (0.5s) = (90,000 / 0.9M / 90M) seconds

= (90,000 / 0.9M / 90M) \* 128/1024 = (11,250 / 112,500 / 11M) GB-s

The first two tier is free. Monthly billable compute GB- s for the third tier is 10.6M

Monthly compute charges = 10.6M \* $0.00001667 = $177

(b). Monthly request charges

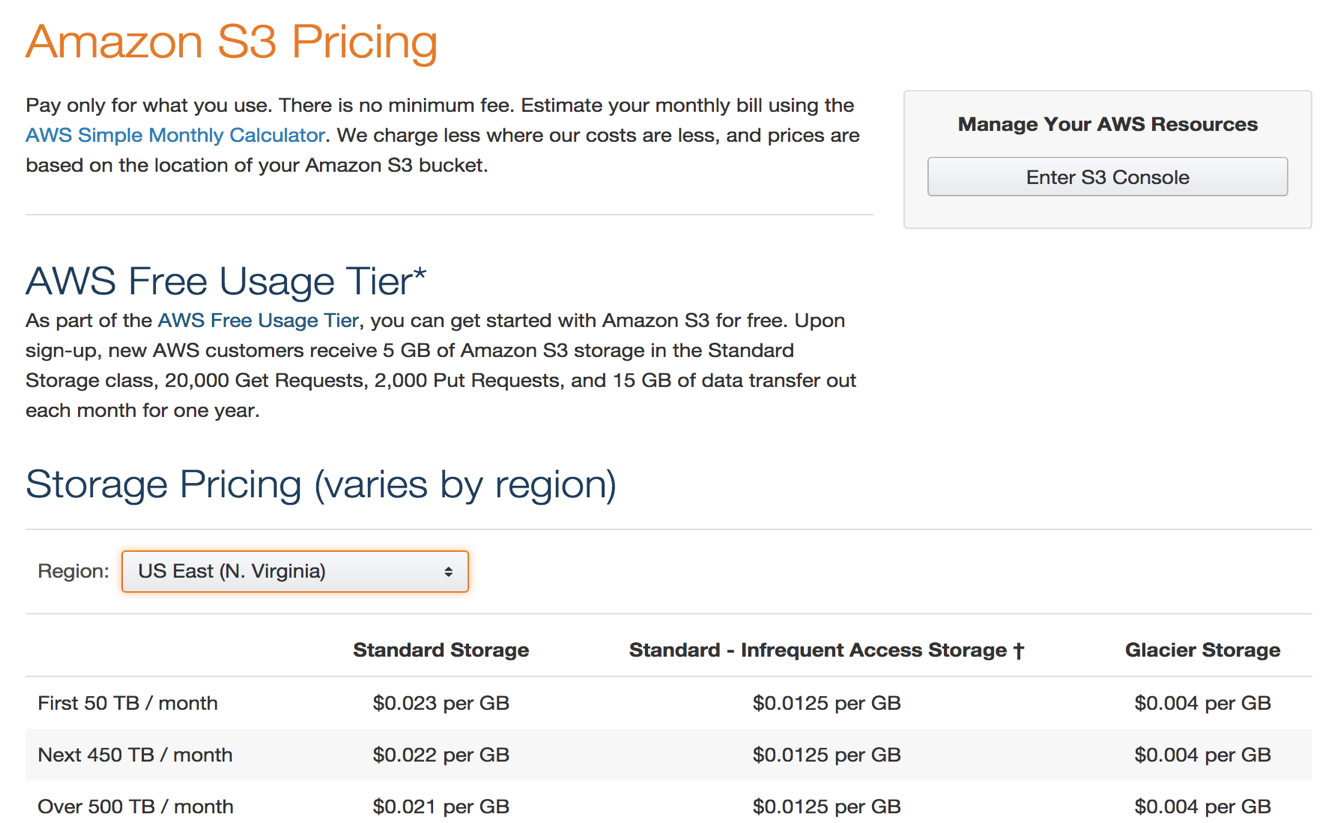
The monthly request price is $0.20 per 1 million requests and the free tier provides 1M requests per month.

Total requests – Free tier requests = Monthly billable requests

Total request charges: Free / 0.8M \* $0.2 / 179M \* 0.2 = free / $0.16 / $35.8

(c).Total charges

Total monthly charges = free / $0.16 / $213

* S3 Storage

Our system uses S3 bucket for two purposes: store images and image scores.

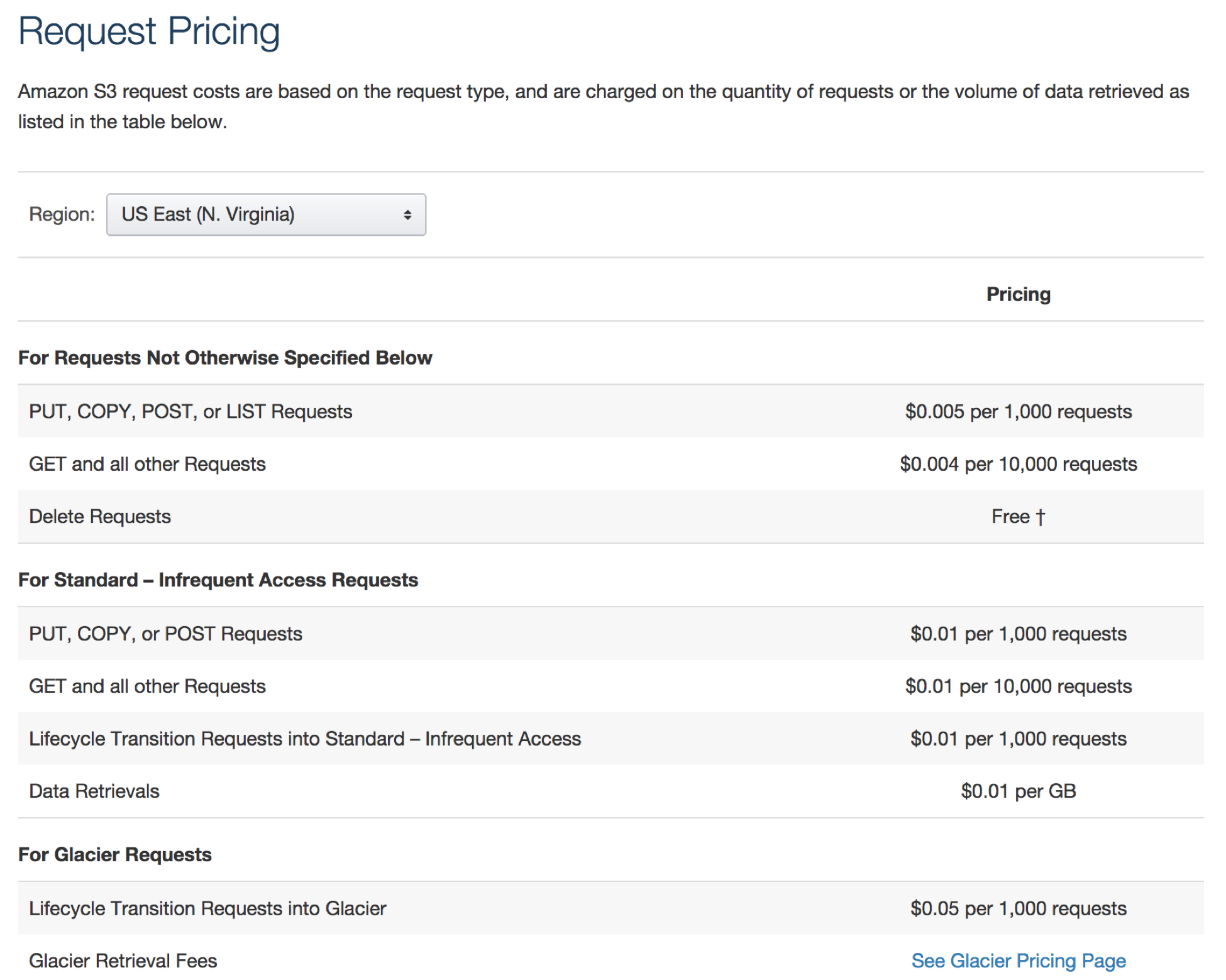
(a). For images

According to our assumption, the average size images is 1MB. So the total storage of images in one month on three tiers would be 88GB / 0.9 TB / 90 TB. Because we will access images in S3 bucket at most once a month, infrequent Access Storage would be enough. Cost: $1.1 / $11.25 / $1125

(b). For image scores

One score (one image) will take 16 bytes memory. One user will have 90 scores in month. The the memory needed for three user-tier would be around 1.4 MB / 14 MB /1.4 GB.

Total cost on S3 storage: $1.1 / $11.26 / $1,125

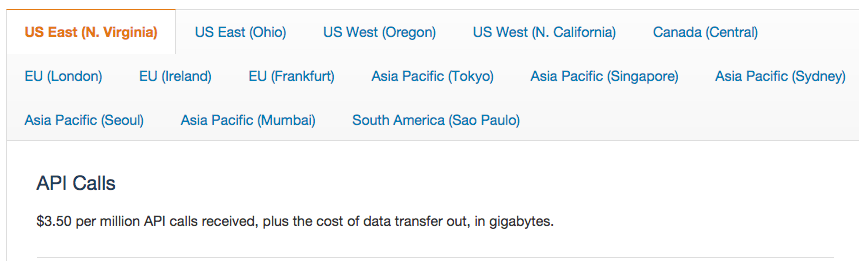
* S3 Request Pricing Charges

We will be using data retrieval service.

For 1,000 user, we need 88GB + 1.4MB, the cost would be $0.88

For 10,000 and 1 million users, it would be $8.8 and $880

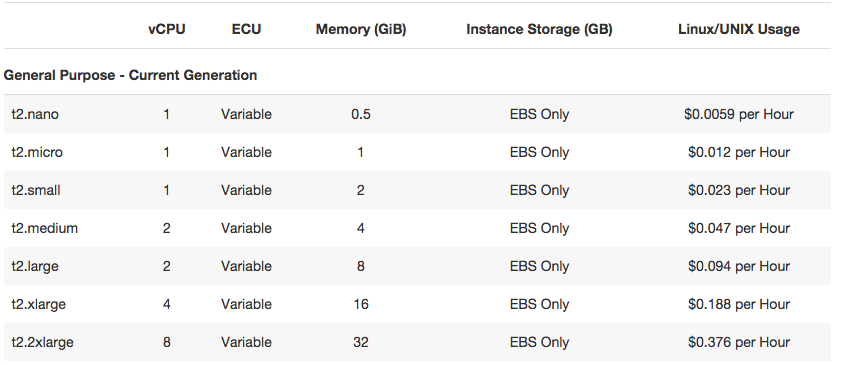
* AWS API Gateway



For one user, our system will make 2 API calls.

Total cost for three user-level tiers: $3.5 / $3.5 / $7.0

* EC2



According to the recommended use cases of EC2 instances, we will use T2 instances for web applications. For 1,000 and 100,000 users, we use t2.nano. When the user population grows to one million, the AWS will execute horizontal scaling and we assume the system will be using a t2.small instance then.

For one month, the instance need to run every second, that’s 2,592,000 seconds.

Cost: $0.0059 \* 2,592,000 = $15,298; $0.012 \* 2,592,000 = $31,104

* Developing - one time fee

We use experienced estimation method. Assume the system is a 1 man-year work, and the maintenance cost each year is 20% of total cost.

The average salary of software engineers is $90,000. Then the developing cost would $90,000.

* Maintenance

Suppose our system will only need considerable maintenance fee when it comes to million-level users. According to the developing cost, the monthly maintenance fee:

$90,000 / 12 \* 0.2 = $1,500

* Labor

Training Fee

Since our system is user friendly, we only need a two-our workshop once a year to teach UPMC staff to use our system if we have 1,000 or 10,000 users. Assume the trainer work 50 weeks a year and 8 hour per day. According to Glassdoor, the average salary for technical trainer is $53,381. Then the hourly wage is $26. Since training fee is a one-time fee, the total cost would be $26 \* 2 = $52 for a year.

Assume UPMC will first launch our system with 1,000 to 10,000 users, and then scale horizontally to 1,000,000 users. We will have another 2-hour workshop then.

Ⅷ. Appendix - Design Document

### System Description

Our system is Smart UPMC insurance evaluation system, which can evaluate client’s health condition scientifically and accurately with only pictures with diet and exercise. Our system is based on AWS service playground, and process all of the pictures in bucket by calling 3 different APIs. When we finished evaluation with one client, the upload bucket will be cleared and waiting for next client’s information. And three different APIs were given different weight value apply into the score. Because some API performs better than other APIs. Besides to this, multiple API can guarantee the normal maintainence of pictures. After detecting the labels of pictures by APIs, we can use our algorithm to judge whether the label appears in the dictionary and if it does, how many scores we should give to this single picture. In the last, we can add all the pictures’ score to get a total one. In addition to food, we also apply exercise, exercise and alcohol into consideration.

1. System Requirement and Assumptions

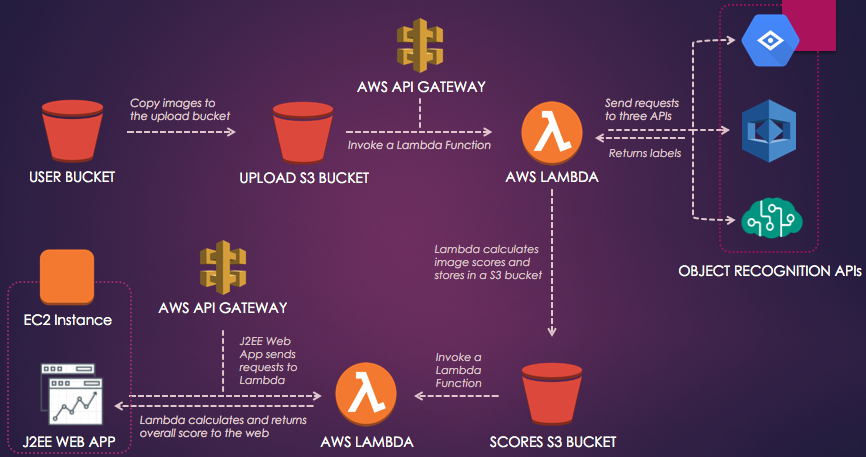
Our system use Python 2.7 as developing language.

All services are based on Amazon Web Services.

For the web application, we use J2EE JAVA 1.8.

Image size is limited within 15MB because we are using AMS S3 service.

1. Architectural Design



1. Tools Evaluation

Functions: AWS Lambda & Google Cloud Functions & Microsoft Functions

|  |  |  |  |
| --- | --- | --- | --- |
|  | AWS Lambda | Google Cloud Functions | Microsoft Functions |
| **Scalability & availability** | Automatic scaling (transparently) | Automatic scaling | Manual or metered scaling (App Service Plan), or sub-second automatic scaling (Consumption Plan) |
| **Max # of functions** | Unlimited functions | 1000 functions per project | Unlimited functions |
| **In-browser code editor** | Yes | Only with Cloud Source Repositories | Functions environment, App Service editor |
| **Pricing** | 1M requests for free, then $0.20/1M invocations, plus $0.000016/GB-sec | 1M requests for free, then $0.40/1M invocations, plus $0.00000231/GB-sec | 1 million requests for free, then $0.20/1M invocations, plus $0.000016/GB-s |
| **HTTP(S) invocation** | API Gateway | HTTP trigger | HTTP trigger |
| **Supported languages** | JavaScript, Java, Ruby, C#, Go, Node.js  PHP and Python | Only JAVA Script | C#, JavaScript, F#, Python, Node.js |
| **Environment variables** | Yes | Not yet | App Settings and Connection Strings only from App Services |
| **Response Time** | =400ms | >400 ms |  |

As the above table shows that although all these three functions are auto-scaling, AWS have several distinctive advantages. For instance: AWS Lambda functions support much more languages than the other two, so it’s much more advantageous for developers who use several programming languages. Also, it has the lowest pricing among these three functions. And the AWS has in-browser code editor, which enable developer develop in the browser and it’s convenient .

Platforms: AWS & Google Cloud Platform & Azure

|  |  |  |  |
| --- | --- | --- | --- |
|  | AWS | Google Cloud Platform | Azure |
| Storage Pricing | $0.03 (standard S3) | $0.03 (standard) | $0.044 (Average) |
| Infrastructure Deployment | Most extensive(more than 40 locations) | Fall behind outside U.S(33 locations) | Fall behind outside U.S(32 locations) |
| Instance Types | 38 | 18 | 33 |

As the above table shows that AWS platform has the lowest storage pricing and also the most extensive infrastructure deployment among the 3 platforms. In this case, the AWS platform enables major company like UPMC health insurance to deploy their service all over the world in more than 40 locations. Besides, because the related instance types of AWS is much more than the other two platforms, it would be much more convenient for the developers to develop related instance in the same platform. In a conclusion, AWS is the best choice of these three platforms so we choose AWS to develop our system.

APIS: Amazon Recognition & Google Vision & Azure Computer Vision

|  |  |  |  |
| --- | --- | --- | --- |
|  | Amazon Rekognition | Google Vision | Azure Computer Vision |
| Pricing | $1 / 1000 events | $1.5 / 1000 events | $1.5 / 1000 Events |
| Average Response Time(*N. Virginia, 1000 files, 10 at a time*) | 1.1s | 0.98s | 2.1s |
| Accuracy | 2st accurate | Most accurate | Less accurate |
| Image Size Limit | 15Mb / Image from S3 | 4 MB / Image | 4 MB / Image |

As the above table shows that Amazon Rekognition has the highest image size limit which will totally expand our service range, because almost all the pictures in the Internet is less than 15 MB. But as for Google Vision and Azure Computer Vision, 4 MB’s limit sets a high requirement for the picture’s quality and size. As for the accuracy, after numerous tests,we discovered that the Google Vision’s recognition is the most accurate and Google Vision has the most diverse and precise labels. Additionally, as for the pricing and average response time, there is not much differences between Amazon Rekognition and Google Vision. So in order to improve the accuracy and performance of our system, we decide to use Amazon Rekognition mainly. To be more precisely, 3 APIs have different weighting factor . For instance, for a picture which worth 1 point, GCP weights 0.8 points because it is the most accurate after numerous testing and the other two APIs only counts 0.1 each, because we use multithreading processing in our system.

1. Tools and Services

* Cloud Computing Platform: Amazon Web Services
* Cloud Function: AWS Lambda Function
* Cloud Storage: AWS S3 Bucket; maybe use DynamoDB as database in future scale
* Virtual Machine: AWS EC2 Service to launch instance and run virtual machines
* API Gateway: AWS API Gateway
* Object Recognition APIs: Amazon Rekognition, Google Vision, Azure Computer Vision
* UI: J2EE Web Application; Highchart

Ⅸ.References

1. [https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/NationalHealthExpendData/NationalHealthAccountsStateHealthAccountsResidence.html](https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-)
2. <https://logz.io/blog/serverless-guide/>
3. <https://cloudacademy.com/blog/google-cloud-functions-serverless/>
4. <https://aws.amazon.com/api-gateway/pricing/>
5. <https://cloud.google.com/vision/pricing>
6. <https://aws.amazon.com/lambda/pricing/>
7. <https://aws.amazon.com/s3/pricing/>