MONASH University
Information Technology

FIT5225 Cloud computing and security
Semester 1, 2020

Assignment 2
TagStore: A Modern Image Storage on the Cloud

Members:	Chenyang Wu	30869811
	Tiantian Lei	30257298
	Yifei Xie	30397022
	Yutian Chen	30223660

I. System design and architecture

1.1 System arrchitecture diagram

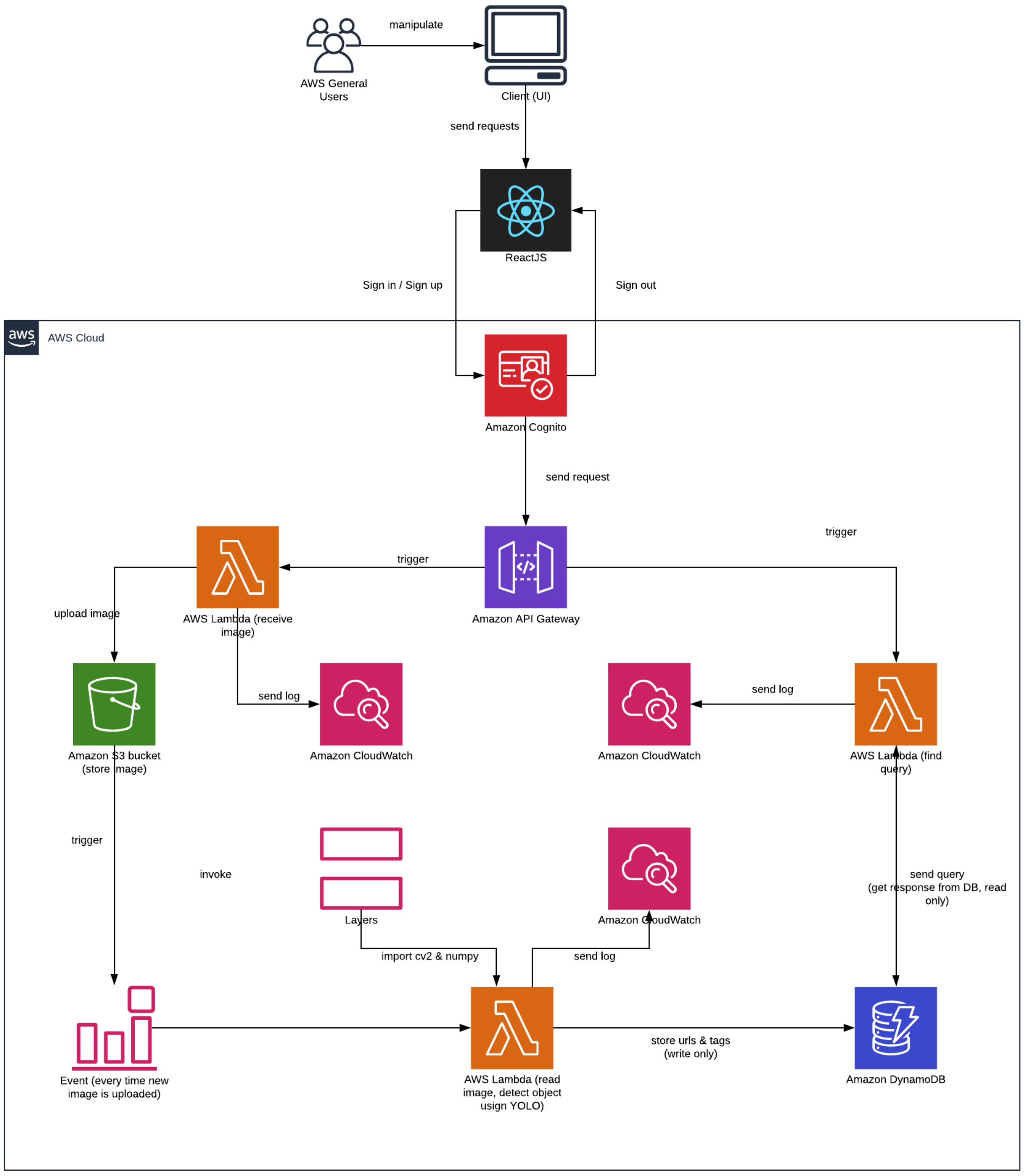


Figure 1 System design and architecture

1.2 Design methodology

This application is a serverless service, which allows users to upload their images to AWS public cloud storage, automatically tags the images. Then the users could query images based on their tags, and get a list of urls of the result images.

1.2.1 Authentication and Authorisation

In this design structure, the application allows users to manipulate a web service as client side to send request to Amazon Cognito. From *Figure 1*, it could be found that except for ReactJS (UI), all the other components are integrated in AWS cloud. Amazon Cognito is a service that protect the web application and internal resources from unauthorised access.

Firstly, a user pool of Amazon Cognito is created for persisting clients' credentials, including their email address as username, first name, last name, and password. This application also provides the function of sending email and reset password whenever the user forget it. Apart from user pool, Amazon API gateway is also connected to Cognito. Different request parameters are set in this gateway, so that only authorised requests could access data.

Besides, roles and permissions are controlled between different components. For example, the lambda function of storing urls and tags is write only to the specific Dynamo DB; and lambda function of finding query is read only. The permission setting follows minimum permission criteria.

1.2.2 Image Upload

After logging in or registration, the user will be navigated to the upload image page and could choose to select files from their local path and then upload them. Once the files are uploaded, a AWS lambda function will trigger, the function of which is to archive the files to the specific Amazon S3 bucket. Simutaneously, the execution logs will be sent to Amazon CloudWatch for debugging and monitoring. There is an event built for monitoring the changes of S3 bucket. If any object is created in the bucket, including PUT and POST, the event is triggered and invoke a lambda function. This lambda function could read the image that just created in S3, then use YOLO to detect the objects in the image, and finally generate an random uid, the url of the image stored in S3, as well as all the detected objects called tags as one JSON object. The uid, url, and tags be then stored in a Dynamo DB. The YOLO lambda function needs some imported library files, like numpy and cv2. These library files are installed in a virtual linux environment and then are uploaded into S3 bucket to form a layer for the lambda function. The logs of YOLO lambda function are also sent to CloudWatch.

1.2.3 Find Query

Another function that user could perform is searching the images they uploaded in Dynamo DB before. In this case, the client will generate a GET request and send it to the API gateway. This GET request will trigger a find query lambda function to read data from Dynamo DB. The function of this lambda function includes receiving a list of tags, then find the corresponding images that have these tags, finally respond a JSON object consists of all the eligible urls. Therefore, the user could receive the response, telling the found image name and accessible url.

1.2.4 User Interface

To help user implement this application more conveniently, the user interface is built through ReactJS. The user interface is created through ReactJS, which is an open-source JavaScript library for building user interfaces. It uses Java Script language to provide a declarative and compoent-based create interactive UI. In our design, clarity is the first and most important job. We have used some light color and significant buttons, so that this application is easy to use. Through interactions between ReactJS and Cognito, the client could implement the following functions, using password and username to login, entering all the required credentials to register, and logging out as well.

II. Team contribution

Member Name	Percentage of Contribution	Description

Chenyang Wu	25%	Cognito authentication;
		YOLO (upload function);
		Complete last 3 parts of the report

Tiantian Lei	25%	Cognito authentication;
		User interface;
		Find query

Yifei Xie	25%	Lambda function using YOLO (Image detection and upload the urls and tags to DynamoDB); Draw the diagram of architecture using Lucidchart; Complete first 3 parts of the report
Yutian Chen	25%	Lambda function using YOLO (Image detection and upload the urls and tags to DynamoDB); Draw the diagram of architecture using Lucidchart; Incorporate the whole codes, configuration, and API

Table 2 Team contribution

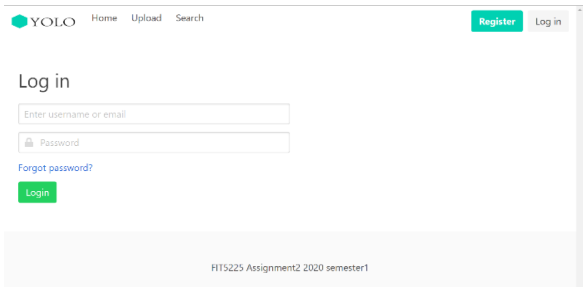
III. User guide

Step 1: Download and extract the zip file from github, the link is provided below in Section IV.

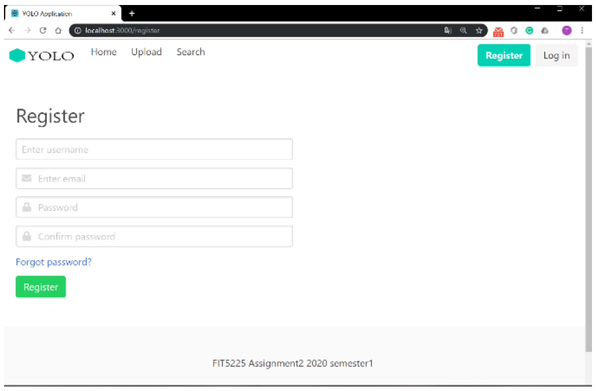
Step 2: Open a command prompt (in window environment) or a terminal, move to the directory where you just extracted the zip file, then enter the following command one bye one:

```
npm i

npm i aws -amplify
npm i tslib -D

npm run start
```

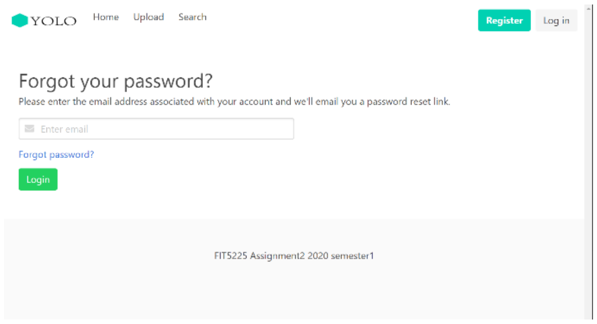
Step 3: After running the application, a new web browser will pop up automatically, showing the home page. Then you could click on login & register. If you forget your password, there is an option that ask for application to send an email to you and reset your password.



FIT5225 Assignment2 2020 semester1



FIT5225 Assignment2 2020 semester1



Please enter the email address associated with your account and we'll email you a password reset link.

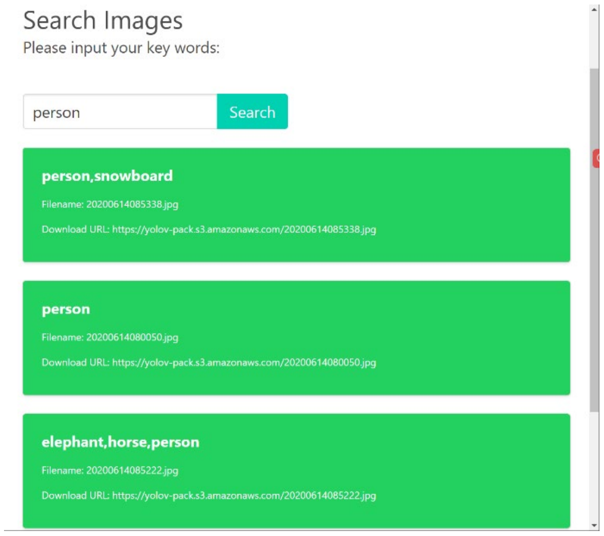
FIT5225 Assignment2 2020 semester1

Step 4: Once login successfully. For uploading images, you need to be in terminal and move to where post.py is (it could be found in lambda_functions/post.py)

```
python post.py -i {image path} -s {filename you want to store in Bucket}
```

e.g.: `python post.py -i 001.jpg -s newImageName.jpg`

Step 5: Here you can change your way to web browser, enter the tag you want to search, and click on search button, a list of response will be seen!


Search Images

Please input your key words:

person

person,snowboard

Filename: 20200614085338.jpg

Download URL: https://yolov-pack.s3.amazonaws.com/20200614085338.jpg

person

Filename: 20200614080050.jpg

Download URL: https://yolov-pack.s3.amazonaws.com/20200614080050.jpg

elephant,horse,person

Filename: 20200614085222.jpg

Download URL: https://yolov-pack.s3.amazonaws.com/20200614085222.jpg

IV. Code link for Github

https://github.com/TiantianLei2020/FIT5225Assignment2-2020-sem1.git

V. Questions

5.1 What are the updates you will perform to your application if you have users from all over the world?

-Improve UI & configuration

We made a detailed user guide in this report. As you can see, at first, users would be required to download our zip file and install node and other necessary command or package to run our simple application. Consequently, we can image that users experience would not be good. If we got more time or money, there are two options to solve this problem.

Utilized Docker. We would pack all necessary packages, libraries, and files into Docker. Once, user downloaded our APP's docker, he could use it directly.

Made this application become a 100 percent web service totally. In this assignment we opened our localhost port to be available for signing in or signing up. The better way to do this is to make a real public website which could be access from outside via using AWS Lightsail service or other similar service. Finally, improve UI to help users upload multiple pics easily.

-Develop Completed Dynamo DB

We would improve our database by setting more query rules and attributes. For example, users should have their own unique ID, so they would be allowed to search their own pictures instead of all related pictures in bucket.

-Display Pictures directly

After querying with API gateway, users would get URLs for wanted pictures back. In our application, if user want to check the content in those images, they could merely download those pictures from S3bucket. Consequently, I proposed that once user click these URLs, the browser would display the pictures.

5.2 What sort of design changes you will make to reduce chance of failures in your application?

-Build back up database & bucket in Multi availability zones

In case of disaster or attack, we should have set up few back up database and bucket. Once our database or function could not work normally, we could use back up bucket and database on time.

-Amazon Machine Image

In case of most users are not familiar with how to use command line to install and configure file, we should help them by using AMI. AMI would contain all essential configuration files, packages, and libraries.

-Deployed Elastic Kubernetes service

In this assignment, we are required to use Lambda to fulfill Function as service. However, based on the limitation, there is no doubt that our service would crash, if at some moment, we got one million requests from users. In order to solve this problem, we could use AWS Elastic Kubernetes service to deploy worker nodes on EC2. Hence, by applying elastic Kubernetes service, we would have multiple worker nodes to do the server simultaneously. We do not worry about the failure anymore.

5.3 What are the design changes you will make to increase the performance of your applications in terms of response time, query handling, and loading images?

-S3 storage and CloudFront

We would choose S3-standard storage service if we have enough cash flow. S3-standard could not only offer highest availability, best durability, multiple available zones but also would charge free for storage and retrieval. We suppose that a lot of clients would barely delete the image. In addition, they would retrieve those pictures. So S3-standard bucket is our best choice.

Moreover, we would like to set up several CloudFront as our CDN. Suppose there are some users may require downloading related pictures after querying. Therefore, CDN would improve the speed of transporting.

-EC2

As I discussed before, Lambda function has a limited performance. For example, the execution time is up to 5 mins, Memory and package zip are also has constraints. Hence, if we use EC2 as our server, then the performance would be improved. If we still cooperate with Lambda, we must adjust default memory and Timeout to support our heavy workload.

-Amazon ElasticCache service

Utilizing amazon ElasticCache service to cache data. Database caching allows users to dramatically increase throughput and lower the data retrieval latency associated with backend databases, which as a result, improves the overall performance of your applications. The cache acts as an adjacent data access layer to our database that ours ’applications can utilize in order to improve performance. A database cache layer can be applied in front of any type of database, including relational and NoSQL databases. Common techniques used to load data.

VI. Use case scenario

Our group all did agree upon that this simple application could did a big favor on some specific industries if we could use this application correctly.

We proposed that we could use this application to explore, investigate, and protect ecosystem.

We would give two examples to demonstrate that our proposal is feasible.

6.1 Explore Ocean eco system

At first, we should collect and obtain as much as ocean creatures’ pictures and data. Training YOLO with those huge date. Make sure that yolo would detect those ocean creatures correctly with low fault toleration. Suppose our application get pictures from a submarine which take a lot of pictures under ocean. Once received photos, YOLO would start reading and analyzing those images. Later on, YOLO would return a specific list of details about detected objects and quantity. Based on the results, we should know a lot of information. For example, we could know the distribution of different species. In addition, YOLO could do the count job. Consequently, we would know that how many animals lived in this specific area. Those data would be meaningful to oenologist, fishing company, and energy corporation. As we know, there are some specific plants would be found if there is a sufficient oil field. Once we find those specific creatures with certain quantity, then energy organizations would be happy to investigate that area.

6.2 Monitor nature ecosystem

Similar scenario, but a little bit different purpose. Human has concurred lands over decades, so we have more completed and detailed animals’ information compared with ocean environments.

We did same step, trained YOLO with fully data. Then suppose, we would get real time information from nature ecosystem. For example, once our application received data from a forest ecosystem, it would read and analysis those data. We think that biologist would be happy to use our application because our application would help them to observe any one ecosystem as they want. Based on results, they can determine animals’ migration routine (think about Kenya). Moreover, they would know the situation of different community. If our application detected that quantity of some certain species decreased sharply, biologist would attempt to save those endangered animals.