ASSIGNMENT 3 REPORT

Linh Dan Nguyen – 103488557
Bao Huy Tran – 103505799
COS20019 – Cloud Computing Architecture
Faculty of Science, Engineering and Technology
Swinburne University of Technology

1. INTRODUCTION

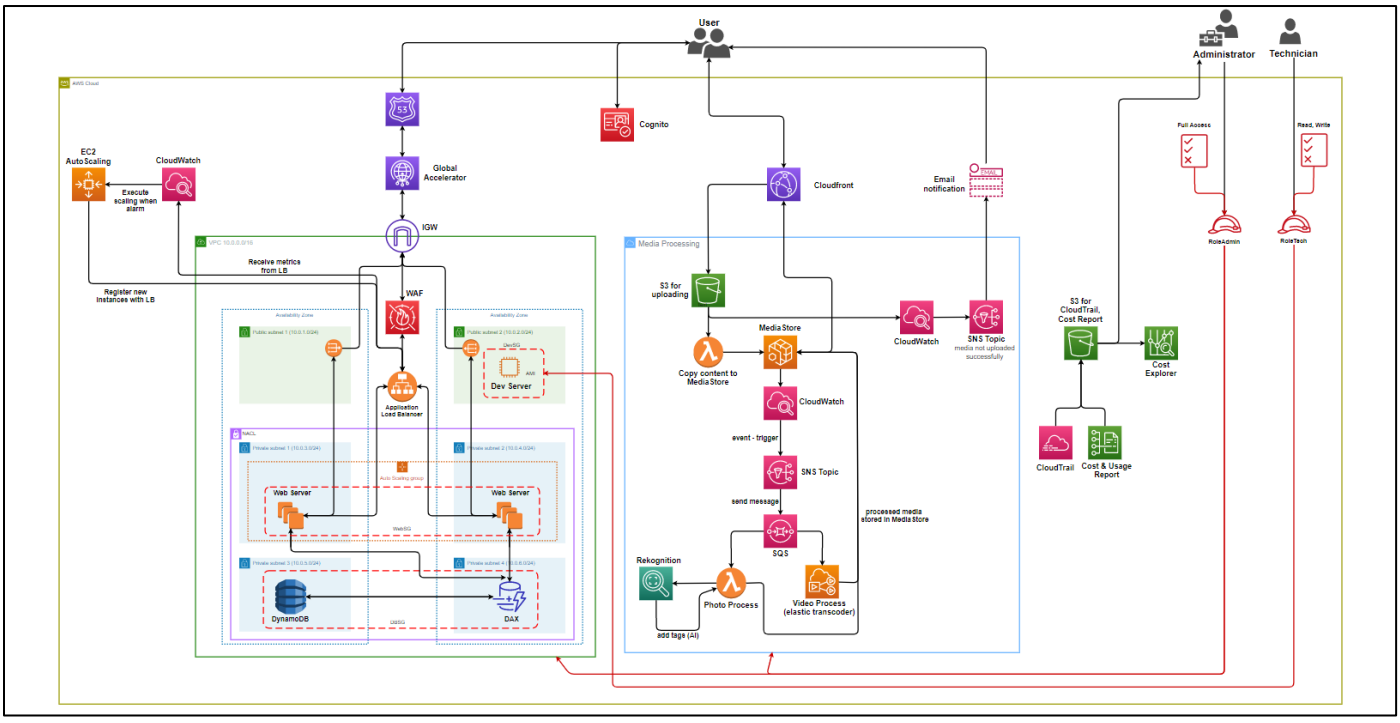
The target of Assignment 3 is to implement more features to the Photo Album website deployed in the previous assignment to help the business achieve more brilliant success. This assignment requires basic knowledge about popular AWS services covered in this unit, and additional services explored by team members. The report will include an architectural diagram, UML diagram to illustrate how business requirements are fulfilled, and the justification of selected services based on the business scenario.

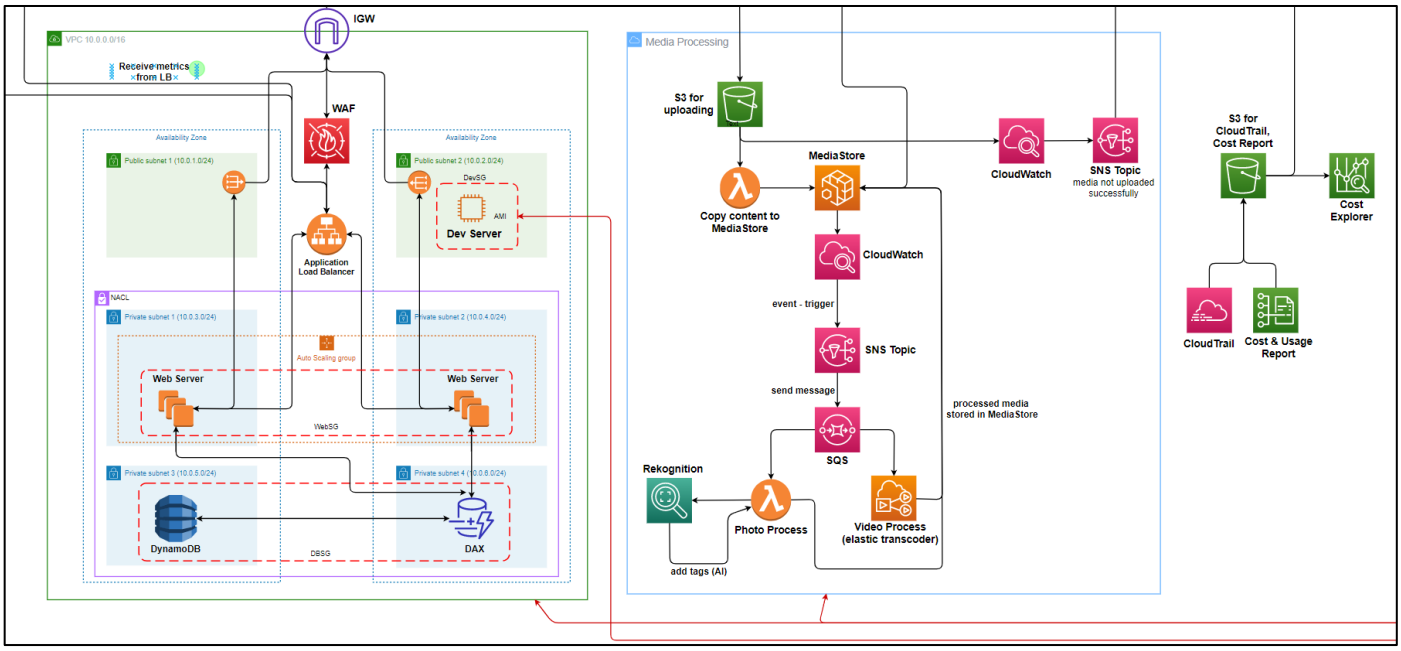
Task allocation:

- Planning and choosing design: Dan and Huy.
- Writing design rationale: Dan.
- Creating diagrams and writing service description: Huy.

2. ARCHITECTURE DESIGN

2.1. Architectural diagram





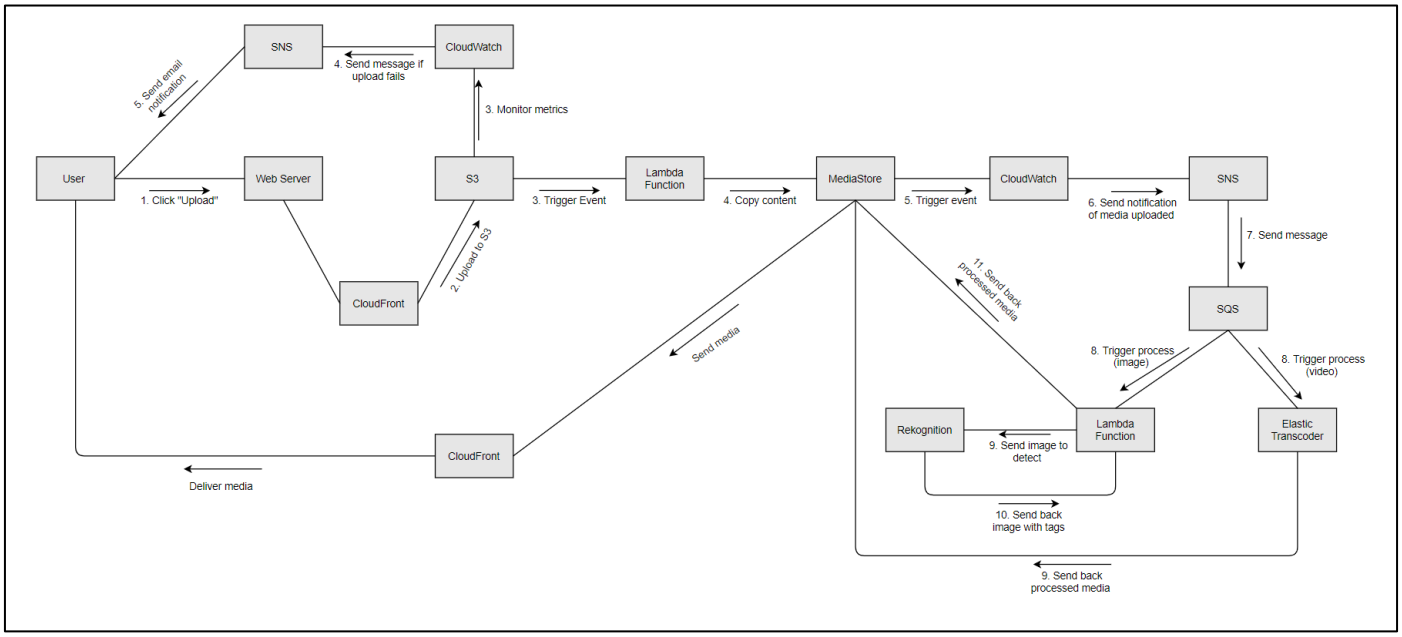
2.2. *AWS Services*

- AWS CloudWatch:
  - Monitor CPU workload and send alarm to EC2 AutoScaling if CPU workload > 60% or <50%.
  - Send event to the SNS topic if the media is not uploaded successfully to S3.
  - Send event to the SNS topic if there is new media copied to MediaStore.
  - Receive metrics from LoadBalancer, S3, MediaStore.
- AWS EC2 AutoScaling:
  - Launch or terminate instances in the autoscaling group based on CloudWatch alarms.
  - Register new instances with LoadBalancer.
- AWS DynamoDB:
  - Provide storage with fast access and high scalability to handle doubling demand.
- AWS DAX:
  - Enable faster response time and data transfer from Web Server to DynamoDB.
- AWS S3:
  - Store the uploaded media from user.
  - Store CloudTrail log files and Cost & Usage report for the administrator.
- AWS MediaStore:
  - Provide highly scalable and durable storage for media files, perfect for handling great demand.
  - Provide cost – effective solution with no up – front costs, no commitment.
  - Integrate with CloudFront to deliver content to user.
- AWS SNS
  - SNS Topic sends email notification if media is not uploaded successfully.
  - SNS Topic send messages to SQS queue about new incoming media to be processed.
- AWS SQS
  - Store the messages from SNS Topic and send the messages to consumers when they are ready.
- AWS Lambda
  - Lambda function to automatically copy uploaded media from S3 to MediaStore.
  - Lambda function to be triggered by SQS's message to perform photo processing (resize,…).

Send resized photo to Rekognition to detect tags, receive photo with tags and store it in MediaStore.

- AWS Rekognition
  - Use deep learning algorithms to perform object detection.
  - Send back photo with tags to the Lambda function.
- AWS Elastic Transcoder
  - Perform video transcoding after being triggered by SQS's message.
- AWS CloudFront
  - Use a global network of edge locations to deliver media to users around the world with low latency and high transfer speed.
- AWS Global Accelerator
  - Route traffic from Route 53 to the optimal endpoints based on geographic location, latency, health check.
  - Improve speed and availability.
- AWS Route 53
  - Route traffic from users to the optimal Global Accelerator endpoints.
- AWS WAF
  - Protect the web application from many types of attack: SQL injection, Cross – site scripting, DDoS.
  - Create customized rule to block dangerous traffic from reaching the LoadBalancer and to the Web Server.
- AWS Cognito
  - Provide user authentication, users have to log in to upload media to the website.
  - Provide user directory to store the credentials of user for authenticating.
- AWS CloudTrail
  - Provide visibility into account activity through the log files.
  - Help with management, troubleshooting, governance.
- AWS Cost & Usage Report
  - Provide customized reports of cost and usage of different AWS services.
- AWS Cost Explorer
  - Access the Cost & Usage report stored in S3.
  - Perform specialized analysis to draw insights from the reports.

2.3. UML diagram of uploading and processing media



3. DESIGN RATIONALE

3.1. How the proposed services satisfy the Business Scenario & Design criteria.

3.1.1.1. Business scenario fulfillment

In this section, I will outline all requirements specified in the business scenario and pose appropriate solutions to address the issues that Photo Album application is facing. The business requirements and suggested AWS services will be summarized in the table below.

Table 1: Solutions to the requirements specified in the business scenario.

Requirements	Solutions
Minimize in-house system management & storage	S3 Bucket: used to store all media content uploaded to the website. S3 is a global storage infrastructure. It provides higher security level, high availability, scalability than physical storage technique. MediaStore: all media files will be copied from S3 to MediaStore to optimize the processing performance for media content. Using DynamoDB to store the media metadata instead of RDS. DynamoDB is a NoSQL database, which supports automatic data replication, unlimited storage, fast data retrieval. It is more cost-saving and flexible than using RDS.
Handle unpredicted growth in the upcoming years	Change instance type from t2.micro to t3.micro for enhanced storage and double CPU power. Create an Auto Scaling group to allow automatic scalability when sudden growth occurs. Auto Scaling will be implemented in combination with CloudWatch and Elastic Load Balancer (ELB). The autoscaling process is event – driven by CPU workload alarms from CloudWatch.

Implement the serverless and event-driven architecture	Lambda: respond to the events, such as changes in DynamoDB, S3, MediaStore. For example, when a user uploads a new photo, it will trigger the Lambda function code to copy the media into MediaStore. S3: an important data storage for server less system. The Lambda code can be triggered when there are changes to the S3 Bucket. MediaStore: the data is copied from S3 to MediaStore in an event – driven manner. CloudFront: a Content Delivery Network, which help the serverless system deliver content to global users effectively with low latency thanks to the edge locations. SNS: can be used to send messages between different parts of the serverless system. It sends massages to other AWS services like AWS Lambda or SQS to trigger the media processing. SQS: allows the system decoupling and microservices scalability. In a serverless application, we can use SQS to manage the flow of messages to divide the work into microservices (i.e. separate processes for photo and video). CloudWatch: used to monitor the performance, health of the serverless system. We can set the alarms to send the notifications to alert when a certain event occurs, or the metrics go out of declared range.
Effective database	The design implemented DynamoDB database system instead of RDS to achieve higher efficiency. Explanantion: - DynamoDB supports automatic data replication. - Highly scalable and flexible. - Higher security level, data encryption. - Better for data recovery than traditional RDS as it supports point-in-time recovery. - Suitable for the applications that requires low latency. - Offers unlimited storage. Besides DynamoDB, we used DAX (DynamoDB Accelerator) is a in-memory cache, which improves the read performance of data in DynamoDB. DAX will create a cache for frequently accessed data to allow later access to the data through cache instead of DynamoDB tables. Photo Album is an application uses database with a high frequency, so implementing DynamoDB and DAX will be beneficial for the app productivity.
Global high availability	There are many AWS services designed to serve the global high availability of applications. The high availability of an app does not limit to the availability of web server, but it also expands to the availability of data storage and database. - Route53: improves the availability of EC2 instances using DNS Failover. Route53 checks the health status of EC2 instances and only route traffic to healthy instances. It can be used to find the closest EC2 instance based on geographic location of users, which could reduce the low latency because of distance, and improves the response time to user's request.

		- DAX: We explained the function of this service in the 'Effective database' requirement. In general, DAX will increase the access performance when users want to read data from the DynamoDB. - CloudFront: reduces the response time when users send the requests to access media content, especially the content stored in MediaStore. This service increases the app availability through edge locations placed around the world. Additional service: AWS Global Accelerator. This service works quite similar to CloudFront. It supports the global coverage, which uses IP address to route traffic to the closest edge location based on the geographic location of users and health status of the endpoint. Users can quickly access the app resources from anywhere in the world.
Media processing	Generate alternative versions for uploaded content	For photo, a lambda function and Rekognition is used to process the image and add tags before storing it back to the MediaStore. For video, Elastic Transcoder is used to transcode videos before storing it back to the MediaStore.
	Use AI to recognize photos	Amazon Rekognition: a service to analyze photos and videos. It can realize human faces, objects and put suitable tags.
	System decoupling	SNS: this service can decouple the system by sending and receiving messages through a pub-sub model. This model helps reduce the system overloading by letting each component processes messages themselves. SQS: decouples the system by performing actions with the messages through a message queue.

3.1.1.2. Design criteria fulfillment

Our team planned to explain the proposed services in the architecture design in two orientations: vertical and horizontal. Table 1 is the design analysis in vertical orientation, we provided the solutions to business issues based on the order of the requirements in the business scenario. In this section, we continue with the horizontal analysis by listing the service functions in 5 pillars of a well-architecture design.

Table 2: How proposed services satisfy the design criteria.

Design criteria (5 Pillars)	How did the design satisfy the criteria.
Performance, Scalability	To deal with unpredicted and high demand, our system implemented two actions: change the instance type, create event – driven auto scaling. The instance type is changed from t2.micro to t3.micro, which is the most suitable option with suitable cost, better storage and CPU power. Event – driven auto scaling is achieved by using CloudWatch to monitor the CPU workload and alarms the EC2 AutoScaling to deploy or terminate instances in the AutoScaling group, the threshold for the CPU workload is from 50% to 60%. To improve performance, we used AWS Cloudfront to reduce the workload on the main server. Cloudfront also supports automatic auto scaling. We changed the database to DynamoDB and used DAX (DynamoDB Accelerator) to improve performance and scalability for the database. Our system also used decoupling design in the media processing section with SNS and SQS to enhance overall performance.

Reliability	The choice of EC2 to host application servers is highly stable and secured. DynamoDB allows automatic data replication across multiple AZs, which ensures reliability. Other services that we chose such as S3, MediaStore, SQS, SNS all can help improve the reliability of the system.
Security	We implemented the layers of security groups and network ACL from the previous design in assignment 2 and improved it by adding additional AWS services. AWS WAF: this service acts as a virtual firewall to protect the application from outside attacks. It can easily find and prevent common types of dangerous attack by filtering incoming traffic to the web application. AWS CloudTrail: records the details of all access to the applications. It can be used to perform policy compliance and increase security. AWS CloudFront: is good at detecting problems and provides prevention methods towards attacks. It has the DDoS protection, SSL/TLS encryption to encrypt the data transfer process between the system and users. AWS IAM: There will be an administrator and a technician in the system. It is important to use the IAM service to create roles for these individuals to guarantee that everyone has the least-privilege to necessary resources. .
Operational Excellence	CloudWatch: keeps track of all metrics related to the app performance to detect the problems (for example: unhealthy instances) and fix it on time. We can create the alarms with threshold to be triggered when needed. CloudFormation: this service is a key to apply 'treat the infrastructure as code' principle. It provides a design user interface to create and manage the infrastructure. It can be used to automate the deployments, prevent errors, and improve the app consistency. The errors can easily occur if we manually deploy everything.
Cost optimization	All of our choices of service is considered carefully to ensure cost optimization (i.e. instance type, NoSQL database, MediaStore). Building an efficient system also contributes to cost optimization. We used certain AWS services to monitor the cost of the system. AWS Cost & Usage Report: This service conducts reports on the cost and usage of the system and sends the report to S3 bucket. AWS Cost Explorer: This service takes the report from the S3 bucket and utilize its powerful built – in functionality to perform detailed analysis, which can help the administrator monitor and make changes if needed Budget: - Fixed expense: employee salary - Variable expense:

	○ Web Server instances (t3.micro): $0.0104/hour ○ DynamoDB: first 25GB free, $0.25/GB/month after ○ MediaStore: $0.023/GB/month ○ S3: $0.023/GB/month ○ CloudWatch: $0.10/alarm metric/month - Some other services have free tier that we can use from now.

3.2. Alternative solutions

3.2.1.1. Virtual machines vs. Containers vs. Serverless computing

	Virtual Machine (EC2)	Containers	Serverless computing
Performance	A virtual machine with complete operating system, runtime, and dependencies. It can execute different types of software and works like a physical machine. However, it is resource-intensive and need a longer time to start compared to other machines. The performance of EC2 instance is quite stable.	A lightweight virtualization box that allows applications to run on a single host system. The container performance can be affected by the container runtime. Crashes in the container runtime can destroy the performance. Containers provide the scalability. Fast start up time.	Allows running code without having a server or managing the infrastructure. The serverless computing has some performance challenges: cold start time, function duration, memory usage.
Reliability	EC2 instance can be deployed in several AZs to increase the fault tolerance.	Not stable. It can easily be affected by multiple factors: container runtime, container image. It supports the app isolation which improve the failure resilience.	It provides the scalability, and low app latency. Serverless computing supports automatic failover and disaster recovery.
Security	You can attach the instances to security groups. Therefore, you can control all security policies in an obvious way.	Better security because the applications and the hosts work independently. However, there are still some security challenges: container sprawl, container breakouts, and container image vulnerabilities.	Suffer from less attacks compared to other deployment methods because there is no server. However, there is not obvious way to secure the serverless model (access to resources, data protection)
Cost	Depend on the instance type, the smaller storage is, the lower the price is.	Depends on the business needs. It saves money on app deployment and management. However, there are additional costs on the Licenses and Infrastructure	No infrastructure code, but there will be costs on the Function code execution (Lambda), data storage cost, and monitoring cost.

Our team decided to use EC2 instance because this is the most stable app deployment method. Although EC2 instance cost is higher than containers and serverless computing model, it deserves for us to pay as it provides a higher security level, the reliability and performance are also guaranteed.

3.2.1.2. SQL vs. NoSQL database

For this architecture, we decided to use DynamoDB (NoSQL database) instead of RDS (SQL database). The differences between two databases will be compared below.

	DynamoDB	RDS
Performance	Allows quick access with low latency using the key-value pattern, it supports the cache to get the frequently accessed data faster. It is more suitable for application with simple data models, and less consistency requirements. This database type does not require the database standby. It also does not require the data to have a clear format.	Slower access to data, but RDS provides a higher consistency level. It is more suitable for application with complex data structure, and the data should follow a defined format. Requires a standby database instance to increase the performance.
Reliability	The best option for auto scaling and fault tolerance. It supports the automatic backup in multiple AZs in a Region.	Supports the automatic backup, high availability in multiple zones. However, it cannot scale automatically like DynamoDB
Security	Provides the encryption at rest and transit, fine-grained access control, VPC implementation and IAM.	The same security features with DynamoDB. RDS has some advanced security features: database auditing, database monitoring.
Cost	Charged per GB storage per month, data transfer will also charge by the GB transferred.	Based on the database instance type. Like EC2, the larger database instance is, the higher cost is charged

We chose to use DynamoDB as it provides better performance than RDS. It is more flexible and allows data to be accessed with low latency. This factor is important as no user wants to wait for their media content processing.

3.2.1.3. S3 vs. MediaStore

	S3	MediaStore
Performance	This is a durable, scalable and high available data storage. S3 offers high performance level for most file formats: backup, media, documents as it supports any data types.	This service is the best storage for media content such as music, photo, and video files. However, it tends to support the content delivery network more than a storage. MediaStore provides the best performance and low latency for media files access.

Reliability	Both provides the data replication and high reliability	
	Suitable for data that needs a scalable storage	Suitable for data that needs a short response time, low latency (data used in the CDN)
Security	Both supports the encryption in transit, and integrate with some security services, such as IAM, KMS, and CloudTrail	
Cost	Both are charged by the amount of data stored, the number of requests made, and the volume of data transferred between services.	

Two data storage services: S3 and MediaStore is also the complementary of each other to deliver media content. In the Photo Album application, when users upload their photos to the website, these photos will be stored in S3 first. Then, all media files will be moved to the MediaStore. Storing media files in the MediaStore will increase the data access efficiency and cut down the response time. We can use S3 solely to store data for content delivery, however, its performance, availability and response time are not as good as using the MediaStore.

3.3. Justification

Our team wants to justify that all proposed services we outlined and discussed in this report can satisfy the growth demand of the Photo Album websites. We selected the services based on the following order: Performance & Scalability, Reliability, Security, and Cost Optimization. Therefore, the app performance is our top priority, we are willing to use some services, which do not have the optimal cost to ensure the high availability and high performance for the Photo Album app. We appreciate the importance of reliability and also concentrate on the app security as we must keep all user's information confidential. Whenever possible, we would choose the service with the most optimized price but offer enough features for the application.

3.4. Recommendations

At the moment, the Photo Album application allows everyone to upload their media files without any management methods. We suggest that the Photo Album application should have a page for sign-in and log-in to allow the app authenticates users and filter which media files belong to which user instead of displaying all media files randomly. Each user will have one folder in the S3 bucket to store all of their uploaded files. The process of building this function comprise of 3 parts:

3.4.1. Setup the Login Page

We have to write one more HTML file for the user interface of the Login page. It will work in a combination with a PHP file. The PHP file will store all information about how the Login page can interact with AWS services, such as S3, MediaStore, Lambda functions, Cognito.

3.4.2 Authenticate users

Services: Cognito

We recommend using the AWS Cognito service to create a User Pool for users' account information storage. The Cognito service setup will follow the steps below:

- Open the Cognito service → Click on Create a user pool
- Configure the user pool settings to manage the password policies (minimum length, maximum length, special characters…), user attributes, and multi-factor authentication (MFA)
- Create an app client, which is the application interacts with the User pool
- Generate & Customize the Login page
- Test the Login page (Finish)

After configuring the Cognito service, users can create their account to upload and manage their media content on the Photo Album application. If the account satisfies all policies declared in the User pool, that account will be accepted, and the Login details will be stored in the User pool.

3.4.3. Setup S3 bucket & MediaStore

First, we have to create a Lambda function to automatically create a new folder in the S3 Bucket with the folder name follows format storage_username. When a user uploads their photos/videos on the application for the first time, the Lambda function will be triggered to create a private folder for that user. In the PHP file, we change the S3 Bucket path to the storage_username format, so it can access the correct uploaded data of each user. We also change the path name in the connection between S3 and MediaStore to help the MediaStore gets data from correct user folder.

4. CONCLUSION

The report has successfully summarized our design architecture and design rational for assignment 3. This assignment is particularly more challenging than previous assignments due to the huge number of researches and evaluations to choose the most suitable services. By working on this assignment, we have gained more knowledge about the vast system of AWS, and we also had the chance to practice team working skill and communication skill. Although our design still contains possible configuration problems due to our lack of real experience, we have demonstrated a decent understanding of different AWS services. There is always room for improvement and this assignment has motivated us to dig deeper into AWS.