COS20019 – Cloud Computing Architecture

Assignment 3

Architecture Design Report

Pham Do Tien Phong - 104189767
Le Quang Ngoc Dung 104323277
Swinburne University Of Technology
Faculty of Science, Engineering and Technology

I. Abstract

Cloud computing is gaining popularity for architectural development. The current cloud system for a photo album web application is operational, but modifications are crucial to meet future user demands. This paper focuses on creating an advanced cloud architecture for the photo album website, utilizing new AWS services. The design aims to enhance extended operations while ensuring key criteria like scalability, reliability, fault-tolerance, performance excellence, and cost-effectiveness.

II. Introduction

Photoalbum web application is a platform which enables users to upload and save their photos. Due to the highly increasing requirements of customers, the system may not adapt all of these, leading to low performance and loss of trustworthiness in clients. To ensure the high availability, a new upgrade will be implemented. The new architecture is designed to be more higher availability, scalable as well as cost-effective, which leverages these modifications so as to provide customers with the best performance. This paper provides architecture design related to UML diagrams and tables presenting total cost. This will be constructed the same as the business scenario to meet the needs in the long term.

<u>Business Scenario</u>

1.Using cost-effective and quick to run database
2.Utilized managed cloud services to reduce in-house system administration and storage
3.Design architecture cope with expected growth
4.Improve global response time
5.Handle video
6.Generate multiple media versions upon media upload
7.Ensure extensible architecture for media processing
8.Ensure the architecture is effectively decoupled to avoid overloading.

III. Architecture Design

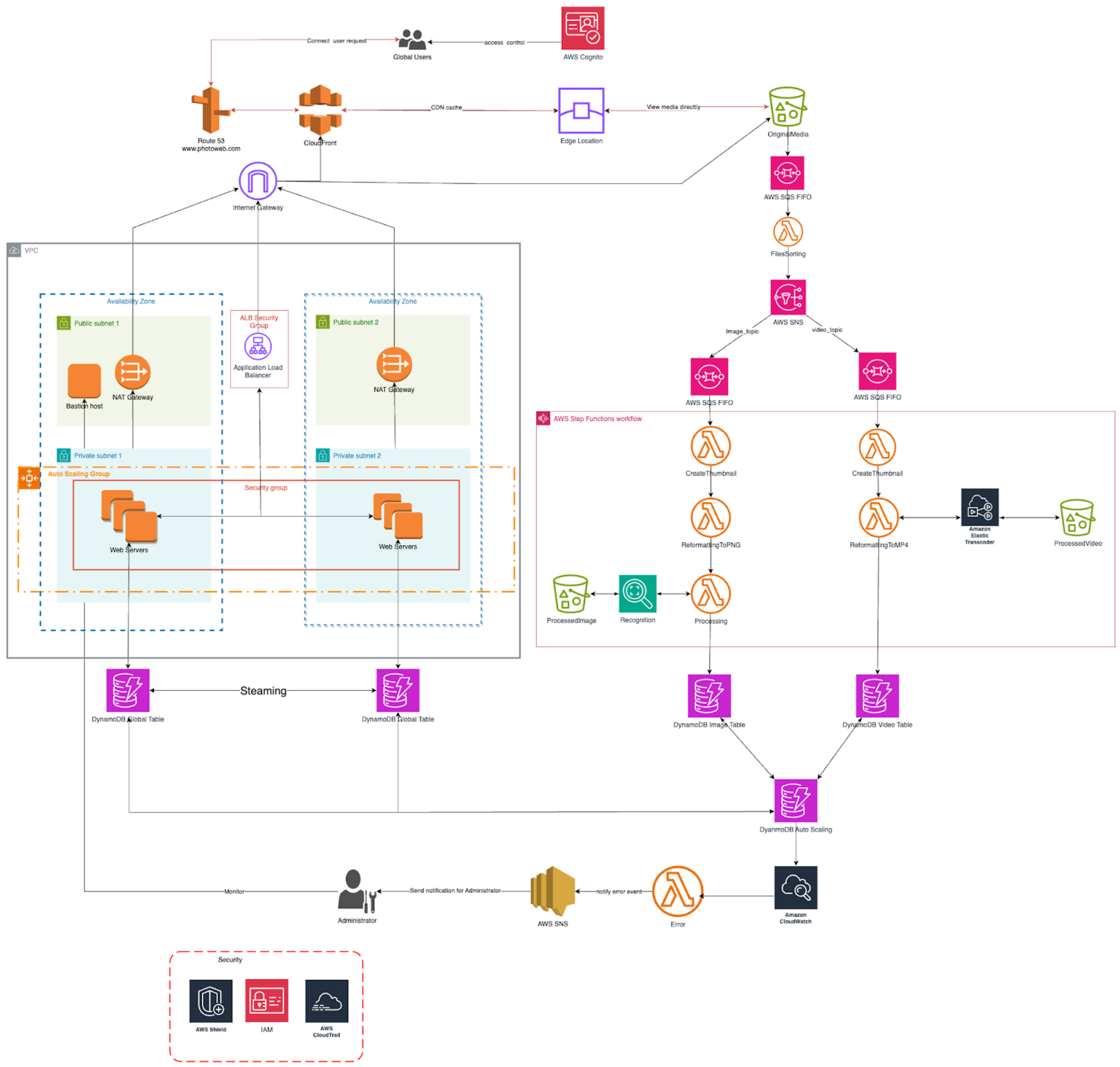


Figure 1. Architecture diagram

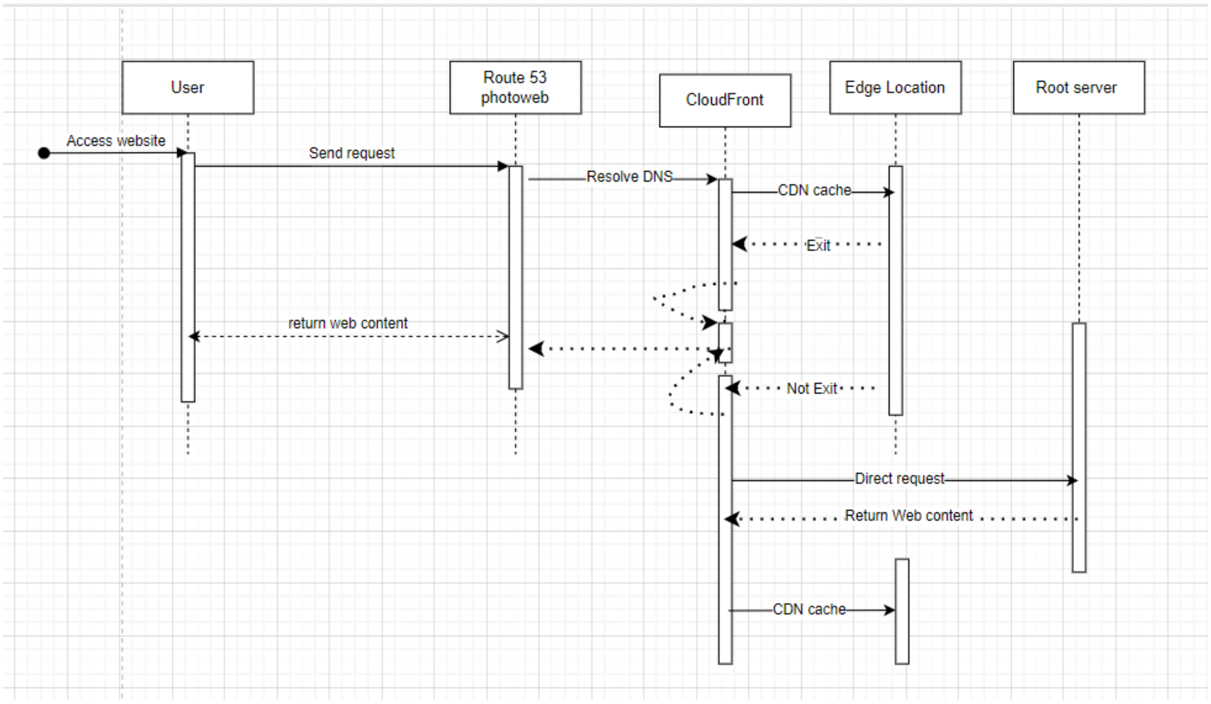
One of the most famous architectural diagrams for cloud-systems is the three-tier architecture. This is why this design pattern will be implemented to upgrade photoalbum web application, composed of three distinct layers:

1. Presentation tier: The initial tier that directly interacts with users. In this website, it is the website where albums of photos are displayed and uploaded to the media of users.

2. Application tier: The middle tier is the most major process in this architecture. This is because this progress is undertaken to upload media files to storage, reformat their metadata into a database and produce new resized versions.

3. Data tier: The final tier is that contains the user storage and database, including media files as well as metadata, respectively.

A. Presentation tier



Firgure 2. UML for display media tier flow

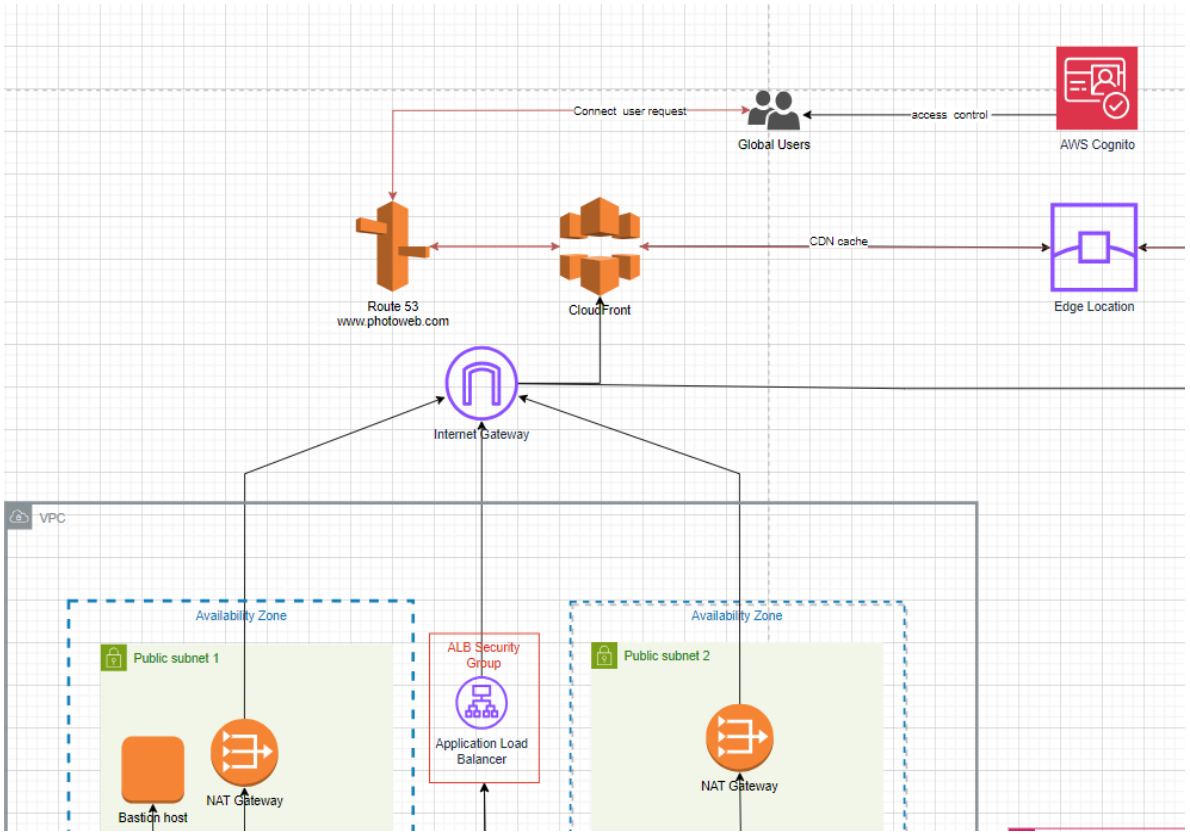
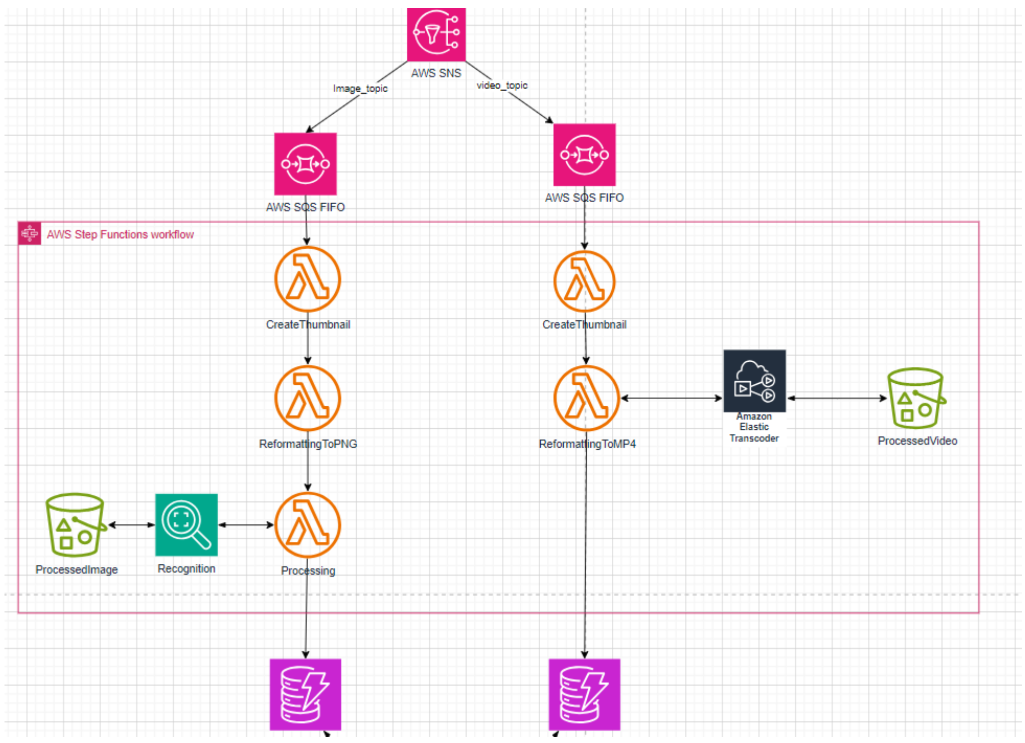


Figure 3.Presentation tier flow

Initially, the static website is stored on Amazon S3 and hosted in Public subnet 1 to monitor the architecture, which will make up for the presentation tier. With our architecture, the 2 most important microservices which keep high availability and scalability of the system are

ELB (Elastic Load Balancing) and Auto-Scaling Group. These services can be customized if needed to adapt the customers demands.
In addition, the company has operated photoalbum web applications to a large number of users around the world, but other regions outside of Australia have resulted in low response time. This is why AWS CloudFront is applied to deliver static and dynamic web content to users at low latency. CloudFront employs a vast network of data centers called edge locations strategically positioned across the globe. These edge locations act as points of presence, enabling the delivery of content to users with minimal latency and optimal performance. When a user initiates a request for content served by CloudFront, the request is automatically routed to the edge location that offers the lowest latency. This selection is based on factors such as network proximity and network conditions, ensuring that the content is delivered from a location that can provide the fastest response time.By leveraging this distributed architecture, CloudFront minimizes the distance between end users and the content they request. This proximity reduces the time delay, or latency, associated with content delivery, resulting in a significantly improved user experience. Additionally, CloudFront employs various caching techniques at the edge locations to further enhance performance. Content that is frequently accessed by users can be stored in the edge location's cache, enabling subsequent requests for the same content to be served directly from the cache, eliminating the need to retrieve it from the origin server. This caching mechanism significantly reduces latency and improves response times.Amazon Cognito streamlines the processes of user authentication, registration, and permissions, resulting in a seamless and secure user experience within the media processing system. Cognito enhances the security of the UI Layer by effectively managing user identities. To ensure efficient domain routing, the UI Layer utilizes Amazon Route 53, which connects user requests to CloudFront and other infrastructure components. This service optimizes traffic flow to AWS resources. Integration is essential to establish a reliable and customizable user interface. The UI Layer further strengthens access security through IAM policies, which restrict unauthorized activities within the media processing system and safeguard sensitive user data and media content.

B. Application tier



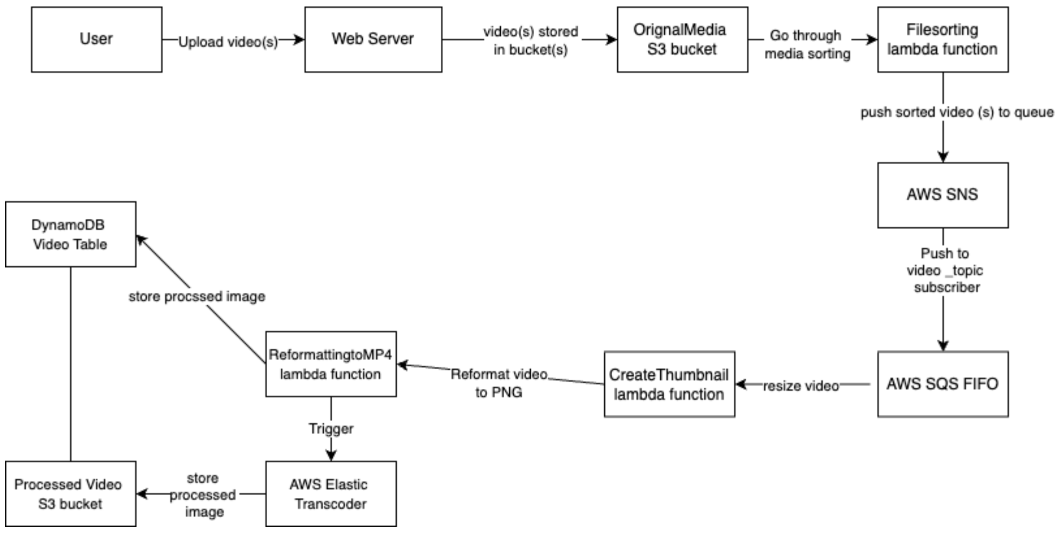


Figure : Upload video UML diagram

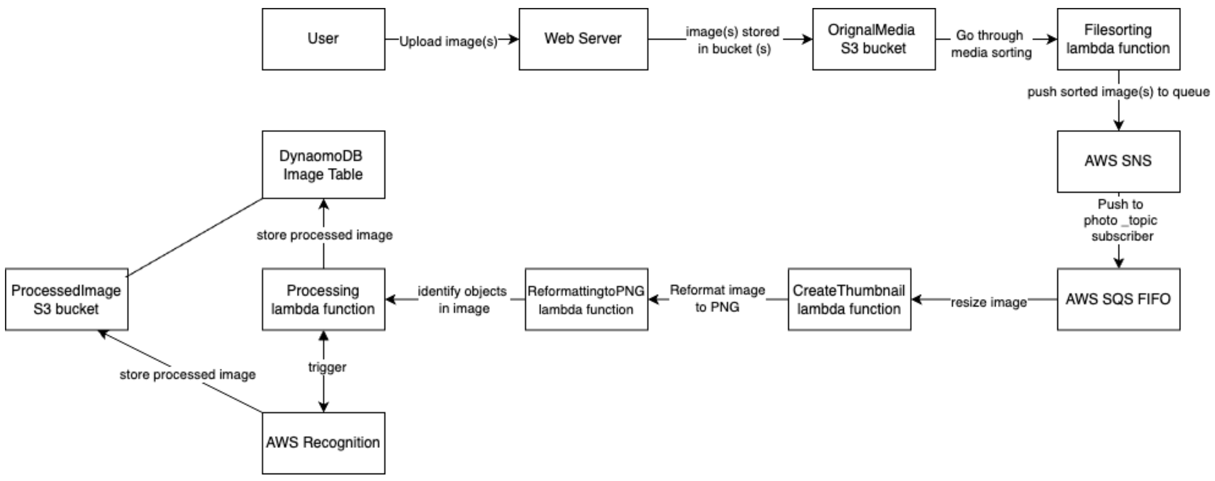


Figure :Upload image UML diagram

In the updated architectural design, the web application retains its functionality for uploading media files and their metadata, similar to the previous design. The process involves uploading the media files to an S3 bucket and subsequently sending the metadata, including the media file's URL, to the CreateThumbnail Lambda function through AWS SDK/API call. The function then inserts the metadata into the database. The significant change in this new design relates to the reformatting and media-transforming function. With the design, the application can adapt requirements of images as well videos. With images, it will be reformatted to .PNG while videos will become .MP4.Efficient storage and retrieval of data are achieved through the utilization of DynamoDB, a highly scalable and high-performance NoSQL database. By integrating with Amazon Simple Queue Service (SQS), the system ensures the separation of components, enabling it to handle increased workloads and system faults by facilitating asynchronous communication between different parts of the system. This architecture promotes resilience and fault tolerance within the system.

Decoupling

To ensure proper decoupling and prevent a single point of failure in the new design, two key services, Simple Notification Service (SNS) and Simple Queue Service (SQS), are utilized. In this architectural design, SNS and SQS are implemented based on the fanout architectural design pattern. This pattern aims to distribute incoming events to different downstream processes, effectively decoupling the system components. Here's how the system operates:

1. The FileSorting Lambda function adds the S3 URL of the file and a key identifying its media type to the message. It then sends the message to SNS.
2. An SNS topic is created, which contains two subscriptions. Each subscription is linked to an SQS endpoint and has a filtering policy to selectively forward image or video messages. This ensures that each type of message is directed to a separate SQS queue and awaits processing by the corresponding Lambda function.

By utilizing SNS and SQS in this manner, the system achieves decoupling between the different workstreams. Issues or interruptions in one workstream do not impact the others, enhancing overall fault tolerance and resilience.

C. Data tier

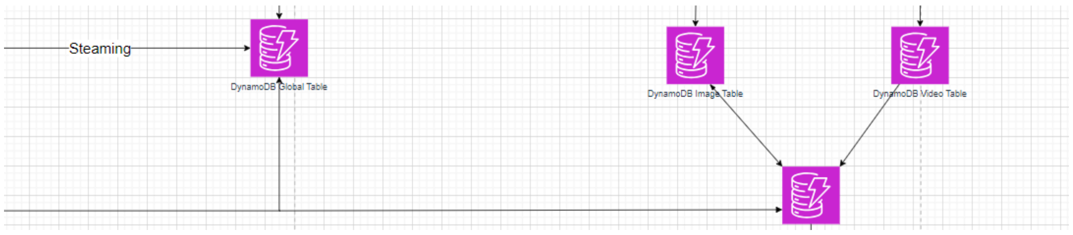


Figure: DynamoDB

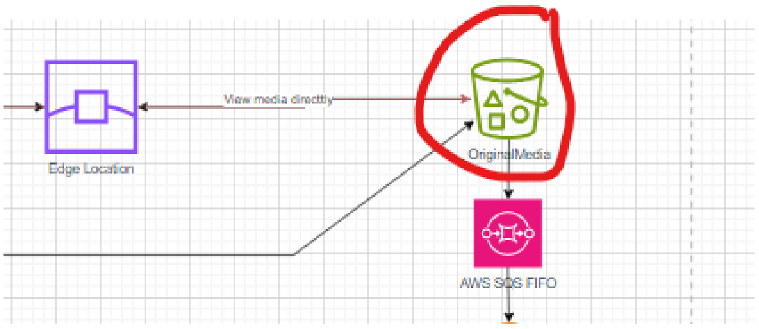


Figure:S3 bucket

The photo album web application currently relies on AWS Relational Database Service (RDS) for storing file metadata. However, the cost-effectiveness of using RDS becomes questionable considering the simplistic table structure of the web application. In light of this, a more viable alternative is proposed wherein the database solution will be migrated to AWS DynamoDB, a NoSQL database service. The adoption of DynamoDB is driven by its inherent ability to horizontally scale, accommodating the rapid expansion of the system, given its uncomplicated database structure.

D. Other UML Diagram

In this section, alongside the architectural diagrams, we provide UML Collaboration Diagrams that illustrate the interactions between users and different AWS services within our cloud architecture. These diagrams offer a comprehensive visual representation of the various use cases, presenting a detailed visualization of how users engage with the AWS services in our system.

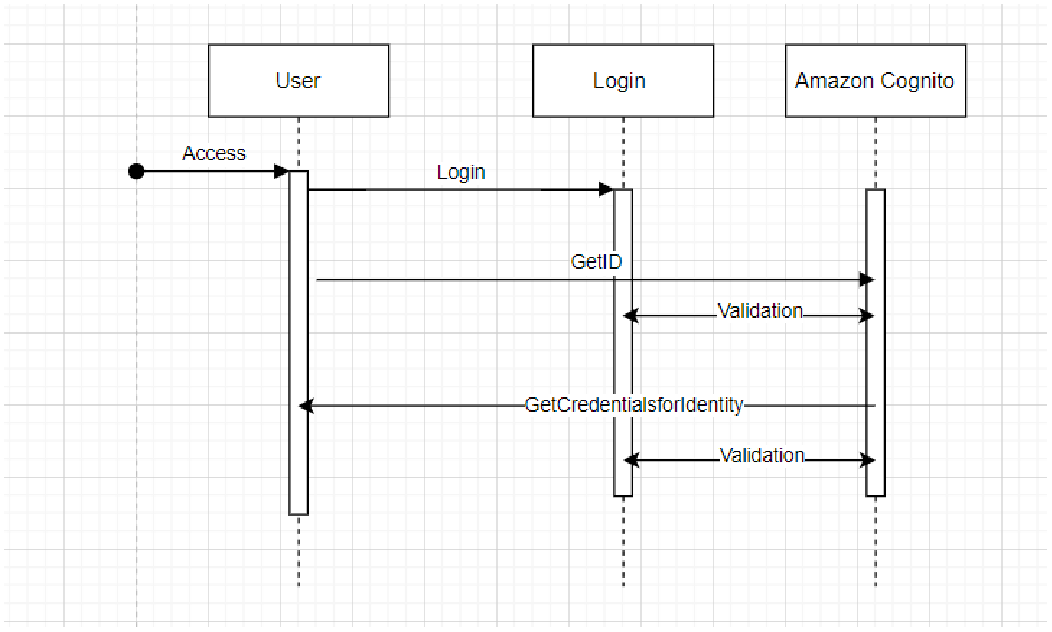


Figure: Sign up & Log in UML diagram

When users access our website for the first time, they are required to complete the signup process. During signup, they provide information such as their name, email, and phone number, which is then transmitted to the web server. This information passes through an SQS queue before reaching the SignUpLogIn Lambda function. The function processes the details and adds new users to the user pool in Cognito. Cognito subsequently sends a verification code to the users through one of the edge locations in CloudFront, enabling faster response times. Users enter the verification code on the website, and it follows the same path as the registration details to reach Cognito for verification against the user pool.
For the login process, users enter their usernames and passwords, which are sent to Cognito for authorization. Upon successful verification, Cognito grants the user an IAM role that allows them to perform necessary actions on the website.

IV. Design Rationale:

A. Business scenario fulfillment

1.Cost-effective and quick to run database
Cost efficiency and resource utilization in media processing are key priorities for the company. To achieve this, the architecture incorporates various strategies such as auto-scaling, the pay-as-you-go model of AWS Lambda, and the tiered pricing of Elastic Transcoder. These measures aim to maximize the utilization of resources while minimizing costs.
By leveraging auto-scaling, the system dynamically adjusts the allocation of resources based on demand. This ensures that resources are efficiently utilized during both high-demand and low-demand periods, preventing overprovisioning or underutilization.
The pay-as-you-go model of AWS Lambda allows the company to pay only for the actual usage of the serverless functions. This cost-effective approach ensures that resources are

allocated and billed proportionally to the actual processing requirements, avoiding unnecessary expenses.
Additionally, the tiered pricing structure of Elastic Transcoder allows the company to select the most cost-effective encoding options based on their specific needs. By choosing the appropriate transcoding tier, the company can optimize resource utilization and reduce costs while still meeting the required media processing requirements.

2.Utilized managed cloud services to reduce in-house system administration and storage
By utilizing managed cloud services in each tier, the cloud system can benefit from reduced in-house system administration efforts. The cloud provider takes care of infrastructure management, maintenance, and scaling, allowing the company to focus on developing and enhancing the web application while optimizing resource utilization and minimizing storage management complexities.

3.Design architecture cope with expected growth
By utilizing services such as AWS Lambda, DynamoDB, AWS CloudFront, AWS Route 53, AWS SNS, and SQS, the system is designed to scale up and down dynamically based on traffic and storage requirements, allowing it to adapt to varying demands. This architecture follows an event-driven approach, leveraging SNS, SQS, and Lambda to effectively decouple the components and achieve scalability and reliability.
When the system experiences high traffic volumes, the Lambda service, and Route 53 can scale up seamlessly to handle the increased load while maintaining low latency and consistent performance. This ability to scale dynamically ensures that the system can effectively handle surges in traffic without compromising its responsiveness.
The event-driven design pattern, facilitated by SNS, SQS, and Lambda, allows for efficient message passing and processing between components. This decoupling of components enables them to function independently, providing scalability and fault tolerance. As a result, the system can handle varying workloads and maintain reliability even during peak usage periods.

4.Improve global response time
The popular Photoalbum application has experienced a decline in performance due to increased usage. To address this issue, a new design is implemented with the support of CloudFront. This design leverages CloudFront's caching capabilities, allowing dynamic content to be stored in edge locations. This caching strategy ensures low latency and maintains consistent performance for users accessing the application.
Additionally, Route 53 is utilized to optimize traffic routing. By directing users to the nearest server, the design aims to enhance the user experience by reducing latency and enabling faster loading of web applications. This improves overall performance and responsiveness for users accessing the Photoalbum application from different geographical locations.

5.Handle video
For handling video uploads, our architecture leverages AWS services seamlessly. User uploads trigger storage in the Original Media S3 bucket through the Web Server. The File

Sorting Lambda Function ensures systematic organization, and AWS SNS efficiently queues sorted videos. DynamoDB's Video Table records essential information, while the Reformatting to MP4 and Create Thumbnail Lambda Functions execute specific tasks. AWS SQS FIFO guarantees orderly processing, and the final processed videos reside in the Processed Video S3 bucket, triggering AWS Elastic Transcoder for efficient transcoding.

6.Generate multiple media versions upon media upload

To fulfill the requirement of generating multiple media versions upon upload, our architecture incorporates four Lambda functions: Create Thumbnail, Reformatting to PNG, Reformatting to MP4, and Processing. These functions work in tandem to process the uploaded media, producing various versions upon media upload. Alongside this, AWS Recognition takes a prominent role in our system, contributing to the intelligent processing of media content. This integration, combined with the capabilities of AWS Transcoder, ensures a dynamic and responsive system capable of generating diverse media versions automatically upon user upload.

7.Ensure extensible architecture for media processing

To ensure an extensible architecture for media processing, our design leverages separate Lambda functions, providing a modular and scalable foundation. This architecture enables straightforward integration of additional processing functionalities in the future without disrupting existing services. Each Lambda function handles a specific aspect of media processing, such as creating thumbnails, reformatting to PNG, reformatting to MP4, and general processing. This modular approach ensures that the system can easily accommodate new processing services or improvements, allowing for seamless expansion and adaptation to evolving media processing requirements.

8.Ensure the architecture is effectively decoupled to avoid overloading.

To ensure an effectively decoupled architecture and prevent overloading, our design incorporates AWS SQS and SNS services. When a user uploads videos, the Web Server stores them in the Original Media S3 bucket. The File Sorting Lambda Function organizes the media and then pushes the sorted videos to an AWS SNS topic. This decoupling allows for efficient communication between components without direct dependencies, ensuring a scalable and resilient system. Subsequently, AWS SNS notifies subscribers, initiating further processing. Before the videos reach the Lambda functions, they pass through two queue services, providing an additional layer of decoupling. This approach ensures that the system can gracefully handle varying workloads, preventing bottlenecks and maintaining optimal performance throughout the media processing workflow.

B. Alternative solutions comparing

1. Application Load Balancer vs API vs Container

While API Gateway and container-based solutions have their merits, they might not be the most suitable options for our upgraded photoweb architecture. API Gateway, designed primarily for managing APIs and serverless functions, may not offer the same level of

versatility needed for our application's diverse requirements. It could introduce complexity in handling tasks like media processing and load balancing.
On the other hand, container-based solutions, while providing a portable and scalable environment, might not align perfectly with our goal of minimizing in-house system administration. Containers require more operational overhead, including orchestration tools and ongoing management. This could counteract the objective of leveraging managed cloud services to streamline our infrastructure.
Furthermore, both API Gateway and container solutions might not be as well-integrated with other AWS services as ALB. ALB's seamless compatibility with EC2 instances and various databases, coupled with its ability to efficiently distribute traffic and handle global load balancing, makes it a more holistic and user-friendly choice for our upgraded photoweb architecture. In summary, while API Gateway and containers have their strengths, ALB's adaptability, integration capabilities, and ease of management make it a better fit for our specific project requirements.

2. **RDS Database vs DynamoDB**

Opting for Amazon DynamoDB over Amazon RDS for our upgraded photoweb architecture offers several advantages. DynamoDB, a fully managed NoSQL database service, excels in providing high performance and seamless scalability, which aligns perfectly with the anticipated growth in demand for our application. Given the simple table structure of our relational database, DynamoDB's flexible schema and automatic scaling capabilities make it a more cost-effective and efficient choice.
Additionally, DynamoDB's global tables feature facilitates improved global response times, a crucial aspect considering our application's widespread user base. Its serverless nature ensures that we only pay for the resources we consume, contributing to overall cost efficiency. DynamoDB is well-suited for handling the potentially vast amounts of media data associated with our photoweb application, offering reliable and low-latency performance.

C.Design Criteria

After thorough evaluation of the AWS services, the subsequent tables will outline the design principles and considerations for each pillar of the well-architected framework.

Category	**Criteria**	**Solution**
Scalability	_Handle large number of traffic _Scale up and scale down when needed	_AWS CloudFront exhibits exceptional scalability and can effortlessly manage a vast number of requests per second. It achieves this through efficient caching mechanisms and load balancing techniques. _AWS Route 53 is adept at handling a high volume of requests, reaching millions per second. It excels at routing these requests from users to the

		closest name server, optimizing latency and enhancing the overall user experience. _Use S3 to scale the storage and retrieval of content. _ Use SNS and SQS to handle requests of customers _ The system is designed to be decoupled enough to modify easily in the future _DynamoDB Auto Scaling continuously monitors workloads and adjusts provisioned capacity in real-time, enabling cost-effective resource utilization without compromising performance.
Reliability	High availability Adapt to changes based on demands Back up data	_ AWS S3 possesses the ability to automatically adjust its scale, increasing or decreasing capacity as per demand. It ensures data durability and replication, providing robust protection for stored data. _ AWS CloudFront exhibits the capability to dynamically scale up or down in response to traffic fluctuations. This scalability feature reduces latency and enhances performance across various regions. Moreover, CloudFront is highly resilient, as it stores multiple copies of content in separate locations, minimizing the risk of failures. _ AWS Cloudwatch can monitor the process _DyamoDB is a scalable NoSQL service which has durable and flexible data with geographical problems.
Security	User data is protected (personal information, uploaded data,..etc) Website prevents unauthorized access and controls authentic access. Data at rest and data in transit are encrypted.	_ AWS IAM is utilized for the authentication and authorization of application access. _ The AWS CloudWatch service can be employed to monitor and identify security threats, as well as detect malware infections. _ AWS S3 offers features such as data encryption, access control, and comprehensive logging capabilities.

		_ AWS Security groups as well as using security at rest, security in-transit. _AWS CloudTrail monitors and guard account activity across the cloud architecture.
Performance	Low latency in every circumstances Fast media upload and download speed Effective storage and retrieval of images	_ AWS CloudFront strategically caches dynamic content in distributed edge locations across the globe, ensuring efficient delivery to users. _ AWS Route 53 intelligently directs users to the nearest and most performant server, optimizing latency and enhancing the overall user experience. _ AWS CloudWatch provides the capability to set alarms for monitoring metrics and triggers automated actions to address any issues that fall outside safe boundaries. _ AWS Elastic Transcoder facilitates the conversion of media files into various formats and resolutions, making them more accessible to users with different devices and connection speeds. - With DynamoDB Global Tables, the active-active replication model allows simultaneous read and write operations in multiple regions, ensuring low-latency access regardless of the user's location.

Cost Estimation for 1 month

Service	**Calculation**	**Total Cost**
Application Load Balancer	1 load balancers x 1.36986 LCUs x 0.0108 LCU price per hour x 730 hours per month = 10.80 USD	10.80 USD
EC2	2059.6426 On-Demand instance hours x 0.011 USD = 22.656069 USD	1483USD

	Dedicated Per Region Fee: 730 hours x 2 USD = 1460.000000 USD	
Route 53	Zone1 has 10,000 records; Zone2 has 20.000 records Host zones : 0.50$/months	401.00 USD
CloudFront	Tiered price for: 1,000 GB 1,000 GB x 0.085 USD = 85.00 USD Total tier cost = 85.00 USD (Data transfer out to internet from United States) Data transfer out to internet cost: 85 USD 1,000 GB x 0.02 USD = 20.00 USD (Data transfer out to origin from United States) Data transfer out to origin cost: 20.00 USD 1,000 requests x 0.000001 USD = 0.00 USD (HTTPS requests from United States))	105.00 USD
Amazon cognito	200 MAUs x 0.10 SAML or OIDC federation requests = 20.00 SAML or OIDC federation MAU requests 20.00 SAML or OIDC federation MAUs - 50 free SAML or OIDC federation MAU requests per month = -30.00 billable SAML or OIDC federation MAU requests Max (-30.00 billable SAML or OIDC federation MAU requests, 0 minimum billable SAML or OIDC federation MAU requests) = 0 total billable SAML or OIDC federation MAU requests SAML or OIDC federation cost (monthly): 0.00 USD	10.00 USD

	200 MAUs - 50000 free MAU requests per month = -49,800.00 billable MAU requests Max (-49800.000000 billable MAU requests, 0 Constant Unit) = 0.00 total billable MAU requests Tiered price for: 0.00 MAUs Total tier cost = 0.00 USD (User Pool MAUs) User Pool MAU cost (monthly): 0.00 USD 200 MAUs x 1 Advanced security features option enabled = 200.00 Advanced security feature MAUs Tiered price for: 200.00 MAUs 200 MAUs x 0.05 USD = 10.00 USD Total tier cost = 10.00 USD (ASF MAUs) Advanced security feature cost (monthly): 10 USD Cognito MAU cost (monthly): 10.00 USD	
VPC, NATgateway	730 hours in a month x 0.045 USD = 32.85 USD (Gateway usage hourly cost) 1,000 GB per month x 0.045 USD = 45.00 USD (NAT Gateway data processing cost) 32.85 USD + 45.00 USD = 77.85 USD (NAT Gateway processing and month hours) 2 NAT Gateways x 77.85 USD = 155.70 USD (Total NAT Gateway usage and data processing cost)	155.70 USD
DynamoDB	DynamoDB data storage cost (Monthly): 250.00 USD Monthly write cost (Monthly): 23.38 USD Monthly read cost (Monthly): 2.76 USD Upfront write cost (Upfront): 150.00 USD Upfront read cost (Upfront): 30.00 USD	276.14 USD

Amazon S3 (Storage)	Total cost (Monthly): 0.01 USD S3 Standard cost (Monthly): 23.00 USD	23.01 USD
CloudWatch	CloudWatch Metrics cost (Monthly): 300.00 USD	300.00 USD
Simple Notification Service (SNS)	Free	0
Simple Queue service (SQS)	1,000 requests per month x 1000000 multiplier for million = 1,000,000,000.00 total standard queue requests Tiered price for: 1,000,000,000.00 requests 1,000,000 requests x 0.00 USD = 0.00 USD 999,000,000 requests x 0.0000004 USD = 399.60 USD Total tier cost: 0.00 USD + 399.60 USD = 399.60 USD (Standard queue requests cost) Total SQS cost (monthly): 399.60 USD	
Elastic Transcoder	100 assets x 10 min x 100 HD output count x 0.03 USD = 3,000.00 USD (HD Transcoding cost per month)	3000USD
Recogniton	Image analysis cost group 1 (monthly): 0 USD Tiered price for: 1,000 API calls 1,000 API calls x 0.001 USD = 1.00 USD Total tier cost = 1.00 USD for images group 2 Image analysis cost group 2 (monthly): 1 USD Image analysis cost for Image Properties (monthly): 0 USD Image analysis cost (monthly): 1 USD Face metadata storage cost (monthly): 0 USD Image pricing (monthly): 1.00 USD	1.00 USD

	Stored video cost (monthly): 0 USD Media analysis cost (monthly): 0 USD Video pricing (monthly): 0.00 USD	
Total Cost/Month		3067 USD