# COS20019 – Cloud Computing Architecture Assignment 3 Serverless/Event-driven Architecture Design Report

Le Hoang Long – 104845140
Faculty of Science, Engineering and Technology
Swinburne University of Technology

Truong Dong Nhi – 104977535
Faculty of Science, Engineering and Technology
Swinburne University of Technology

Nguyen Thien Phuoc - 105028625
Faculty of Science, Engineering and Technology
Swinburne University of Technology

**Abstract**

**This paper presents a serverless, event-driven architectural design for a rapidly growing photo album application. Leveraging AWS cloud services, the proposed solution addresses scalability, performance, and cost-effectiveness challenges while minimizing in-house system administration. Key components include Amazon S3 for storage, AWS Lambda for serverless computing, Amazon DynamoDB for database management, and AWS Step Functions for workflow orchestration. The architecture incorporates global content delivery through Amazon CloudFront and utilizes services like Amazon Rekognition for media processing. This design aims to meet increasing demand, improve global response times, and provide a flexible foundation for future enhancements such as video support.**

## 1. Introduction

The proliferation of digital media and cloud technologies has led to a surge in online photo-sharing and storage applications. As these platforms gain popularity, they face unique challenges in scalability, performance, and cost management. This paper addresses the architectural redesign of a successful photo album application that has experienced rapid growth and anticipates continued expansion. The existing system, which relies on conventional EC2 instances, has reached its maximum performance capacity and faces challenges in achieving satisfactory global response times. To address these issues and prepare for future growth, we propose a serverless, event-driven architecture leveraging various AWS services. This approach aims to minimize in-house system administration while providing a highly scalable and cost-effective solution. Our design incorporates services such as AWS Lambda for serverless computing, Amazon S3 for media storage, DynamoDB for database management, and AWS Step Functions for coordinating complex workflows. We also utilize Amazon Rekognition for advanced media processing capabilities, and Amazon CloudFront for improved global content delivery, and other AWS services.

This paper will detail the proposed architecture, explain the rationale behind our design choices, and discuss how this solution meets the client's requirements for scalability, performance, and future extensibility. We will also explore alternative solutions and provide a comparative analysis to justify our final design decisions.

## 2. Business scenario

The Photo Album program has achieved significant popularity and needs more development to suit the growing demand. The following requirements and difficulties have been discovered, and it is necessary to take actions to address these issues in order to enhance the customer experience:

a. **Infrastructure Management:**
   Use Amazon managed cloud services to reduce the need for in-house administration.
b. **Scalability:**
   The architecture must be designed to accommodate future demand that may increase unpredictably.
c. **Serverless/ Event-Driven Architecture:**
   The company intends to transition from EC2 servers to serverless and Event-Driven solutions.
d. **Cross-Region Performance:**
   The application has been widely adopted globally; however, its response time in nations other than Australia has been rather sluggish. There is a need to enhance global reaction times.
e. **Database Optimization:**
   The performance of the relational database is comparatively sluggish, and the operational expenses are high. Considering the uncomplicated table arrangement, the organization desires to investigate other choices that are more economical.
f. **Media Processing Extension and Optimization:**
   There are plans to expand the system's capabilities to include video media in the future. Users have the ability to upload media in many formats. The company desires the automated generation of many versions of material, such as thumbnails, low-resolution versions ideal for mobile phones, or video transcoding. The process of reformatting, transcoding, or reprocessing material must adhere to many requirements.

## 3. Architectural Design

### a. Architecture Design Diagram

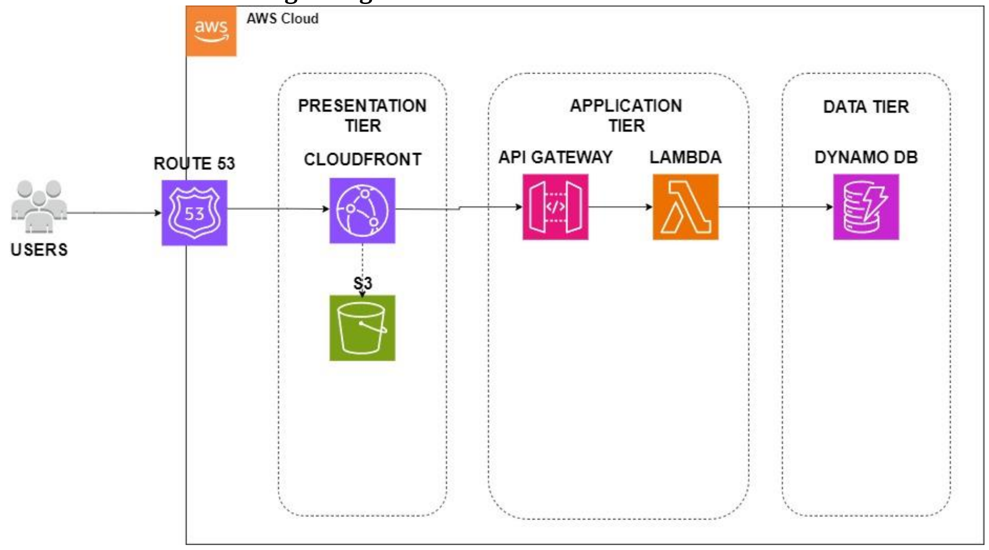

*Figure 1. 3-Tier Serverless Architecture Diagram Design*

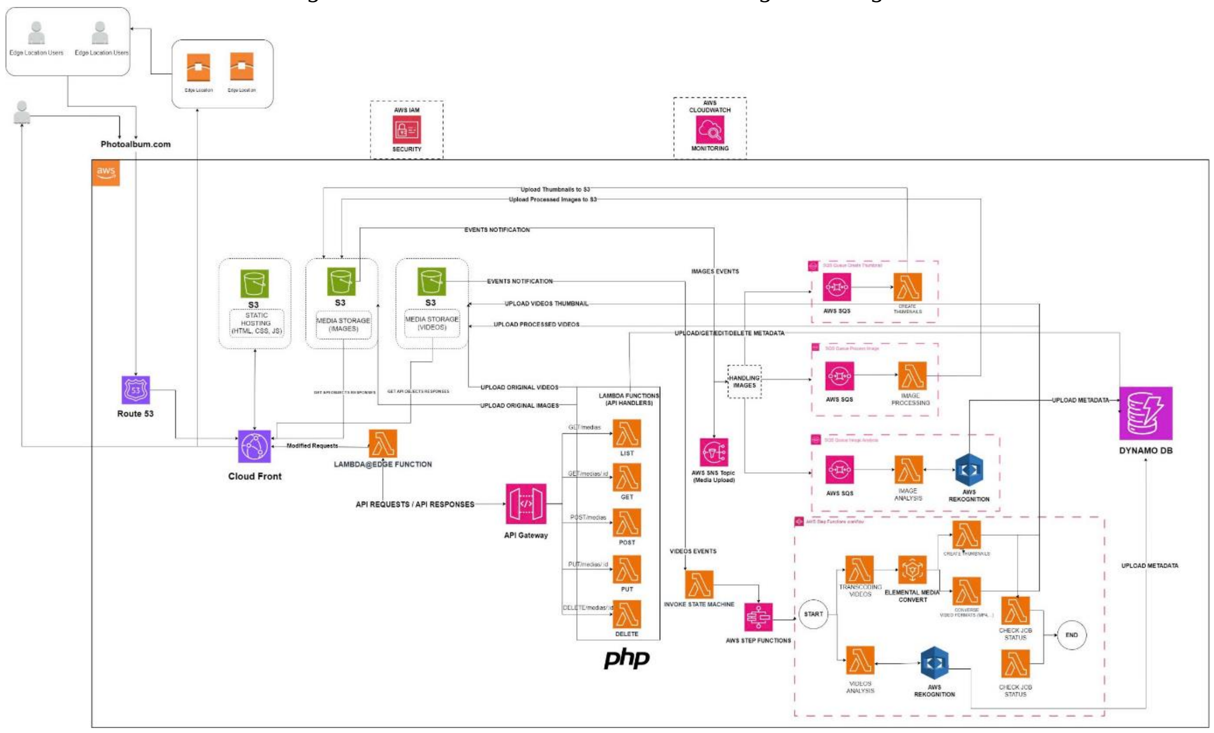

*Figure 2. AWS Architecture Diagram Design*

This architecture adheres to the AWS 3-Tier Architecture, which consists of three separate layers: the Presentation Tier, the Application Tier, and the Data Tier. The architecture can be further subdivided into four smaller components according to the business requirements.

### b. Detailed System Design Solutions

#### i. Global Content Delivery Service

The increasing popularity of the website needs a correspondingly effective content delivery mechanism in order to provide an optimal user experience. This may be accomplished by utilizing AWS Route 53, a DNS solution developed by AWS with a focus on high availability and reliability, together with Cloudfront, an AWS Content Delivery Network designed for optimal speed, security, and developer ease.
Users accessing the website experience different flows based on their location and the content's cache status. Origin location users connect directly to CloudFront,

which retrieves content from the S3 bucket, offering quick access for those near the primary infrastructure. Users at edge locations without cached data trigger a process where their request is forwarded to CloudFront, content is fetched from S3, cached at the edge, and then delivered. This initial request may be slightly slower, but subsequent ones improve. For users at edge locations with cached data, the experience is fastest, as content is served directly from the edge cache without contacting CloudFront or S3. This tiered system optimizes delivery speed and reduces origin server load, with CloudFront acting as the central, intelligent distributor of content.

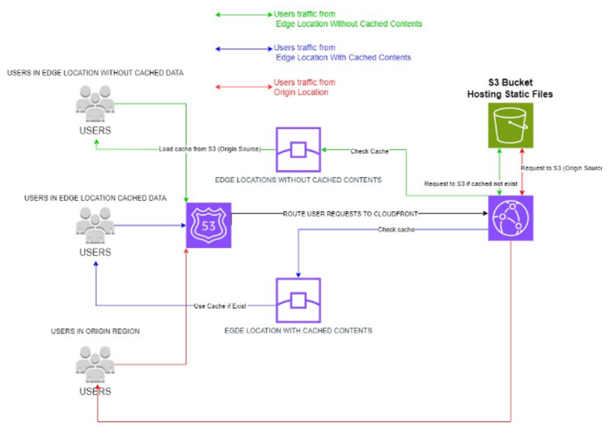
Users traffic from Edge Location Without Cached Contents
Users traffic from Edge Location With Cached Contents
Users traffic from Origin Location
USERS IN EDGE LOCATION WITHOUT CACHED DATA
USERS
S3 Bucket Hosting Static Files
Load cache from S3 (Origin Source)
Check Cache
Request to S3 (Origin Source
Request to S3 if cached not exist
EDGE LOCATIONS WITHOUT CACHED CONTENTS
USERS IN EDGE LOCATION CACHED DATA
USERS
ROUTE USER REQUESTS TO CLOUDFRONT
Check cache
Use Cache if Exist
USERS IN ORIGIN REGION
EGDE LOCATION WITH CACHED CONTENTS
USERS

*Figure 3. AWS Route 53 and CloudFront Content Delivery Network flow*

## ii. Static Assets Hosting

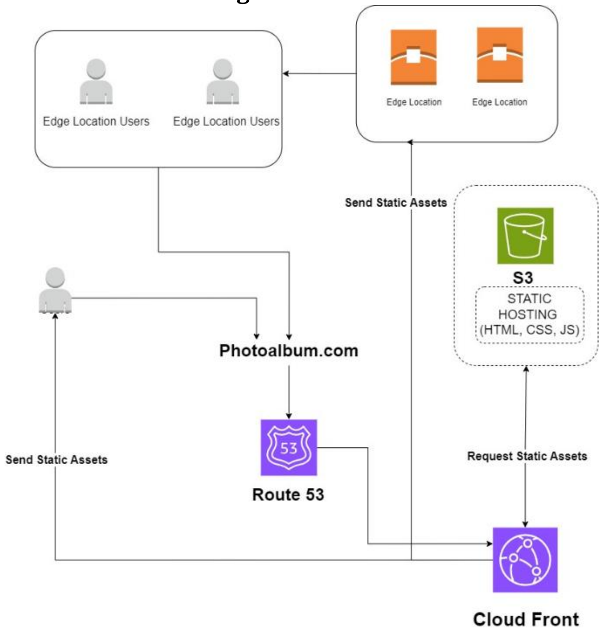

*Figure 4. AWS Route 53 and CloudFront Content Delivery Network flow*

In the previous architectural design, the hosting of static assets, such as HTML and CSS, was done on EC2 servers. The company's administration team had to take control of these servers and implement auto-scaling rules while keeping an eye on health metrics. The new design will move the responsibility for hosting static assets to Amazon Simple Storage Service (AWS S3), a managed service offered by AWS. This change aims to decrease administrative expenses and better correspond with the serverless architecture that the firm intends to deploy.

## iii. Backend application to generate dynamic contents

API Gateway and Lambda functions will manage the dynamic and server-rendering contents of the websites, as well as other business logic written in PHP. The website currently has two primary business logics. The first logic involves retrieving and displaying uploaded material. The second logic allows users to post media, which includes additional procedures for processing those media assets.
In the context of display media logic, Lambda functions will be utilized to invoke the GET/medias endpoint from the API Gateway. Subsequently, the API Gateway will submit a query to DynamoDB to retrieve all the necessary metadata, including the URL link for the media stored in S3. Finally, this information will be transmitted to CloudFront for the purpose of display.
In order to enable users to upload media files, a Lambda function will initiate a POST/medias request to the API Gateway. The original versions of these media files will then be sent to S3, along with information such as name and keywords, which will be stored in DynamoDB. Once the S3 has received the media files, it will send a signal to AWS SNS to initiate image processing. Additionally, it will send a signal to another Lambda function to trigger a state machine in Step Functions. This state machine will coordinate various processes, including the use of AWS Elemental Media Convert for video processing.

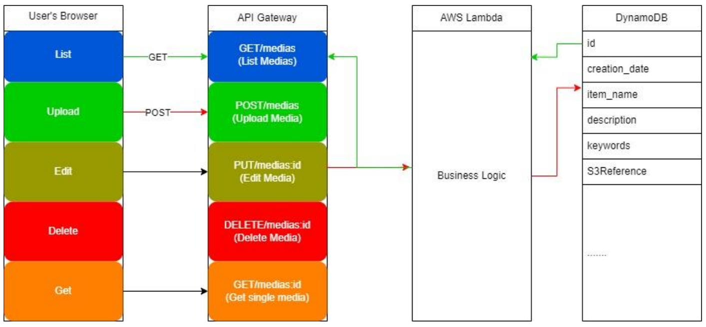

*Figure 5. API Gateway UML diagram*

## iv. Media Processing

### 1. Fan-out architecture

The fan-out architecture is a widely used architectural pattern in AWS that allows for efficient distribution of data to several downstream services or systems.

This design utilizes an event-driven approach, where a single event is triggered and then sent to several receivers. More precisely, this system achieves its goals by utilizing Amazon Simple Notification Service (SNS), Amazon Simple Queue Service (SQS), Amazon Web Services (AWS) Step functions, and Lambda.

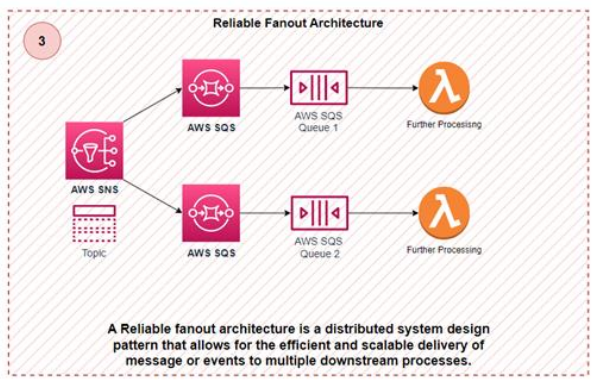
Reliable Fanout Architecture
3
AWS SNS
Topic
AWS SQS
AWS SQS Queue 1
Further Processing
AWS SQS
AWS SQS Queue 2
Further Processing
A Reliable fanout architecture is a distributed system design pattern that allows for the efficient and scalable delivery of message or events to multiple downstream processes.

*Figure 6. Fan-out architecture*

## 2. Media Processing using fan-out architecture

Upon completion of the upload of the original version of the media, the Media Storage S3 Bucket will trigger a notice to activate AWS SNS. This will facilitate the dissemination of communications to a certain subject. Afterwards, AWS SQS is used to arrange the messages in a queue, while AWS Lambda is applied to manage and execute the messages. This architectural approach facilitates the segregation of application components, offers the possibility of scalability, and ensures reliability.

After the media is added to a processing queue, it will be sent to AWS SQS for media processing. If the media type is video, a Step Function will be activated to handle the processing activity.

Regarding media in the form of photos, once SNS receives a notice from S3, it will transmit a signal to numerous SQS instances in order to activate the SQS Queue in a fan-out manner. The SQS queues will concurrently execute the responsibilities of generating thumbnails, processing various variants of the original picture (such as low-resolution and mobile versions), and subsequently uploading these new versions back to S3.

If the media type is video, a lambda function will receive the signal from S3 and then create a state machine to invoke AWS Step functions. The Step function will be configured to run in Parallel mode, enabling the execution of various types of jobs, such as video processing using AWS Elemental Convert. This process can also run in parallel to create thumbnails and convert videos into different formats.

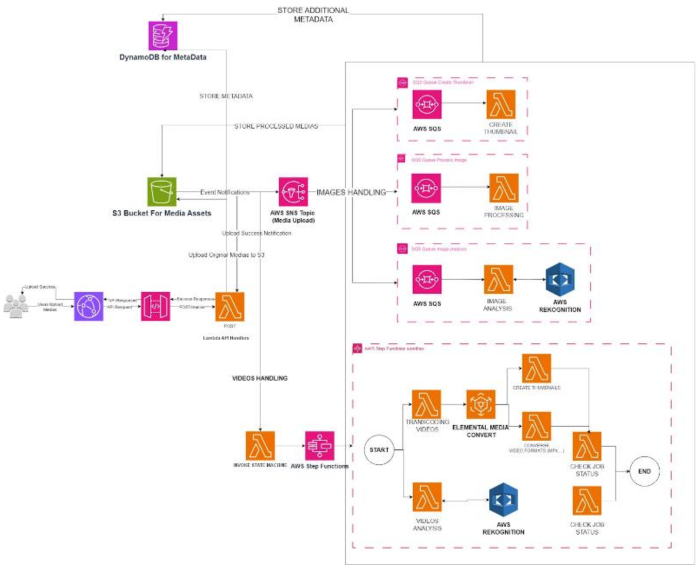

*Figure 7. Media Processing Diagram*

## 4. Design Rationale

### 4.1. Business Scenario Fulfillment

Our proposed serverless, event-driven architecture addresses each of the following business requirements:

#### a. Managed cloud services

Our design leverages a suite of AWS managed services, significantly reducing the need for in-house system administration. We've chosen Amazon S3 for robust and scalable object storage, eliminating concerns about capacity planning or hardware failures. AWS Lambda handles compute tasks without server management, while Amazon DynamoDB provides a fully managed NoSQL database service. By utilizing Amazon CloudFront for content delivery and AWS Step Functions for workflow orchestration, we've created a system that allows the company to focus on application development rather than infrastructure management.

#### b. Scalability

To address the anticipated doubling of demand every 6 months, our architecture incorporates multiple layers of scalability. AWS Lambda automatically scales to handle increased request volumes without manual intervention. DynamoDB's on-demand capacity mode adjusts throughput based on actual usage, ensuring the database can handle growing data volumes and request rates. CloudFront's global edge network scales content delivery as user numbers increase worldwide. Additionally, the event-driven nature of our design, using services like Amazon SQS and SNS, allows for seamless handling of varying workloads, ensuring the system can grow smoothly with demand over the next 2-3 years.

#### c. Serverless/event-driven solution

Our architecture fully embraces the serverless and event-driven paradigms. AWS Lambda forms the backbone of our compute layer, executing code in response to events without server management. API Gateway serves as the entry point for HTTP requests, triggering Lambda functions as needed. We've implemented event-driven workflows using AWS Step Functions, allowing complex media processing tasks to be broken down into manageable, loosely-coupled steps. This approach enhances system flexibility and resilience, allowing individual components to scale and evolve independently.

**d. Cost-effective database**

To address the issues with the existing relational database, we've transitioned to Amazon DynamoDB. This NoSQL solution is particularly well-suited to the application's simple table structure. DynamoDB's on-demand pricing model ensures that costs scale directly with usage, eliminating the need for upfront capacity planning. Its consistent single-digit millisecond latency, regardless of scale, addresses the performance issues of the previous solution. Furthermore, DynamoDB's built-in caching layer, DAX, can be easily added if needed to further boost read performance without significant architectural changes.

**e. Global response time**

To improve response times globally, we've implemented a multi-pronged approach. Amazon CloudFront serves as our content delivery network, caching content at edge locations worldwide to reduce latency for users outside Australia. We've also leveraged Lambda@Edge to move certain compute operations closer to end-users, further reducing response times. DynamoDB Global Tables can be implemented to provide multi-region, multi-master replication, ensuring low-latency database access worldwide. This combination of services ensures that users experience responsive performance regardless of their geographic location.

**f. Video media support**

While the current focus is on image processing, our architecture is designed with future video support in mind. We've incorporated AWS Elemental MediaConvert into our media processing workflow, which can handle a wide range of video transcoding tasks. The modular nature of our Step Functions workflow allows for easy integration of video-specific processing steps when needed, without disrupting existing image processing capabilities.

**g. Media processing**

Our media processing solution meets all specified criteria:

- Automatic triggering: We use S3 event notifications to automatically initiate processing workflows when new media is uploaded.
- Extensibility: Our Step Functions workflow is designed modularly, allowing easy addition of new processing steps, such as AI-based tagging, without major architectural changes.
- Platform flexibility: Different processing tasks leverage appropriate AWS services. For instance, basic image resizing uses Lambda, complex video transcoding uses MediaConvert, and future AI tagging could use Amazon Rekognition.
- Decoupling: We've implemented Amazon SQS queues to decouple media upload from processing. This allows the system to handle upload spikes gracefully, queuing tasks for processing without overwhelming the system. Different queues can be set up for various processing tasks, allowing for specialized worker nodes (implemented as Lambda functions) to handle specific operations like video transcoding or image reformatting.

### 4.2. Alternative Solutions

In developing our serverless, event-driven architecture for the photo album application, we carefully evaluated several alternatives for each major component. This process ensured our final design best meets the company's needs while optimizing performance, cost, scalability, and ease of management.

**a. Virtual Machine vs Serverless:**

- VM-based solution (e.g., EC2): Provides more control over the underlying infrastructure but requires significant management overhead for scaling, monitoring and securing the instances. The initial architecture used EC2 instances to host the website in a private subnet accessed via a NAT gateway.
- Serverless (Chosen): Enables the application to be built and run without provisioning or managing servers. Provides automatic scaling, high availability, and a pay-per-use billing model. The current architecture leverages serverless computing extensively with services like AWS Lambda, Amazon S3, API Gateway, etc. This aligns better with the requirement to minimize in-house administration.
- However, AWS Lambda has a limitation of a maximum runtime of 15 minutes per execution. This means that Lambda may not be suitable for tasks that require more than 15 minutes to complete. In the context of the photo album website, this limitation is not a significant issue, as the current architecture does not include any tasks that exceed the 15-minute runtime limit. Note: AWS Lambda has a maximum runtime of 15 minutes per execution, which may not be suitable for tasks requiring longer execution times. However, this limitation does not affect the photo album website, as no tasks exceed the 15-minute limit.

**b. SQL vs NoSQL database**

- SQL (e.g., MySQL): The relational database used initially was relatively slow and costly to run. It required manual capacity management.

- NoSQL (DynamoDB chosen): Provides a scalable, highly performant and cost-effective NoSQL database solution. DynamoDB's simple data model is well-suited to the application's needs. Its on-demand capacity mode automatically scales and aligns costs with usage. This meets the requirement for a more cost-effective database.

**c. Caching options**

- Application-level caching (e.g., ElastiCache): We considered using ElastiCache, a fully managed in-memory caching service, to improve the performance of the application. ElastiCache could store frequently accessed data in memory, reducing the load on the database. However, after evaluating the performance characteristics of DynamoDB and the caching capabilities of CloudFront, we deemed application-level caching unnecessary. DynamoDB provides consistent single-digit millisecond latency at any scale, which is sufficient for our use case. Moreover, CloudFront caches content at the edge locations, effectively serving as a caching layer.

- CDN caching (CloudFront chosen): We chose to use Amazon CloudFront, a global content delivery network (CDN), to cache and deliver content with low latency and high data transfer speeds. CloudFront's caching capabilities significantly improve response times for users worldwide. By serving content from the edge location nearest to the user, CloudFront reduces the load on the origin servers and provides a better user experience. It also offers advanced features like Lambda@Edge, which allows custom logic to be run at the edge locations for further optimization.

## 4.3. Design Criteria Analysis

**a. Performance and Scalability**

- The serverless architecture enables automatic scaling to handle varying loads efficiently. AWS Lambda automatically scales the number of function instances based on the incoming requests, ensuring optimal performance without manual intervention.
- Amazon DynamoDB, a NoSQL database, provides consistent single-digit millisecond latency at any scale. It seamlessly handles increased traffic and data storage needs, making it highly scalable.
- Amazon S3, used for storing media files, offers virtually unlimited scalability. It automatically scales to accommodate growing storage requirements without any performance impact.
- CloudFront, a content delivery network (CDN), ensures fast content delivery to users worldwide. By caching content at edge locations, CloudFront reduces latency and improves the overall performance of the application.
- The decoupled architecture, using Amazon SQS for message handling, allows for independent scaling of different components. The upload process and media processing can scale separately based on demand, preventing bottlenecks.

**b. Reliability**

- AWS managed services, such as Lambda, S3, DynamoDB, and CloudFront, offer high availability and fault tolerance out of the box. These services are designed to provide reliable performance with minimal downtime.
- The serverless architecture eliminates the need for managing individual server instances, reducing the risk of server failures impacting the application.
- DynamoDB automatically replicates data across multiple Availability Zones (AZs) within a region, providing built-in fault tolerance and data durability.
- S3 offers 99.999999999% (11 9's) of durability, ensuring that data is protected against hardware failures and other disruptions.
- SQS provides reliable message queuing and delivery. It ensures that messages are processed in the correct order (for FIFO queues) and handles message visibility timeouts and dead-letter queues for failed processing attempts.

**c. Security**

- AWS Identity and Access Management (IAM) is used to manage access to AWS resources securely. IAM allows fine-grained control over permissions, ensuring that each component has access only to the resources it needs.
- API Gateway acts as the entry point for all API requests, providing a secure and managed gateway. It integrates with AWS IAM and Amazon Cognito to enforce authentication and authorization for API calls.
- CloudFront offers SSL/TLS encryption for data in transit, ensuring secure communication between users and the application.
- S3 provides server-side encryption for data at rest, protecting stored media files from unauthorized access.

**d. Cost Analysis**

- The serverless architecture enables a pay-per-use pricing model, where costs are incurred only when the resources are actively used. This eliminates the need to pay for idle server time, making it cost-effective for applications with variable workloads.

- DynamoDB's on-demand capacity mode automatically scales based on the application's traffic, aligning costs with actual usage. This prevents over-provisioning and reduces costs during periods of low demand.
- S3's pricing model is based on the amount of data stored and the number of requests made, making it cost-effective for storing large volumes of media files.
- Lambda's pricing is based on the number of requests and the duration of execution, ensuring that costs are proportional to the actual usage of the application.
- CloudFront's pricing model includes data transfer and HTTP/HTTPS requests, which can be optimized by leveraging caching and minimizing the transfer of unnecessary data.

We have analyzed the cost of running the website for the next 2-3 years, considering the expected doubling of demand every 6 months. Three scenarios with total media upload sizes of 50GB, 100GB, and 200GB were examined.
The estimated monthly costs are:

- 50GB: $36.82
- 100GB: $49.55
- 200GB: $113.81

These costs cover the Content Delivery Network, Web architecture, and media processing procedure. Detailed breakdowns can be found in Appendices D, E, and F.
The proposed serverless architecture offers cost-efficiency and scalability, allowing the application to handle growth without significant upfront investments. Costs remain proportional to usage, and the architecture dynamically scales resources as needed.
Regular cost monitoring using AWS Cost Explorer and setting up budgets and alerts are recommended to maintain control over expenses.
In summary, the proposed architecture provides a cost-effective solution that accommodates the anticipated growth over the next 2-3 years while offering the necessary technical capabilities and cost optimization benefits.

## 5. Conclusion

Ultimately, implementing a serverless/event-driven architecture on AWS may significantly advantage businesses in need of a scalable and cost-efficient solution. AWS services such as AWS Lambda, Amazon SNS, and Amazon SQS can automatically handle real-time events, eliminating the need for human administration. This allows company owners to allocate more efforts towards delivering value to their consumers.

## 6. Appendix

**Appendix 1: Service used in the architecture**

### a. Amazon S3 (Simple Storage Service)

- Functionality: Amazon S3 supports scalable, durable, and secure object storage. It is used to store in the database and retrieve data from the end-consumers..
- Why it is used:
  - Scalability: Automatically scales to accommodate the storage needs as they grow.
  - Durability: Ensures high durability and availability for stored data through redundancy across multiple devices and facilities.
  - Cost-Effectiveness: Offers a pay-as-you-go pricing model, reducing storage costs.
  - Justification: Meets requirement #1 for using managed cloud services, specifically for storing media files in a scalable, durable, and cost-effective manner.

### b. Amazon CloudFront

- Functionality: Amazon CloudFront is a fast content delivery network (CDN) service that securely delivers data, videos, applications, and APIs to customers globally with low latency and high transfer speeds.
- Why it is used:
  - Improved Performance: Distributes content closer to end-users using a global network of edge locations, thus reducing latency.
  - Security: Provides DDoS protection and integrated with AWS Shield for additional security.
  - Justification: Addresses requirement #6 by enhancing global response times and ensuring secure content delivery.

### c. AWS Lambda

- Functionality: AWS Lambda is a serverless compute service that allows you to run code without provisioning or managing servers. It automatically scales your application by running code in response to events.
- Why it is used:
  - Serverless Architecture: Eliminates the need to manage infrastructure, reducing operational overhead.
  - Cost-Effective: Charges only for the compute time consumed.
  - Event-Driven Processing: Ideal for media processing tasks triggered by S3 events, API Gateway requests, etc.
  - Justification: Meets requirements #1 and #4 for using managed services and implementing serverless solutions for media processing.

### d. Amazon DynamoDB

- Functionality: Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.
- Why it is used:
  - Performance: Low-latency responses for high-throughput applications.
  - Scalability: Handles large amounts of data and high request rates without manual intervention.
  - Cost-Effective: Pay-per-use pricing model is more economical compared to traditional relational databases.
  - Justification: Addresses requirement #5 by offering a more cost-effective database solution suitable for the application's needs.

### e. Amazon SQS (Simple Queue Service)

- Functionality: Amazon SQS is a fully managed message queuing service that enables you to decouple and scale microservices, distributed systems, and serverless applications.
- Why it is used:
  - Decoupling: Separates media processing tasks to ensure each component can operate independently and efficiently.
  - Reliability: Ensures message delivery even if the receiving component is temporarily unavailable.
  - Justification: Meets requirement #8d by providing a reliable and managed message queuing service to decouple media processing tasks.

### f. AWS Step Functions

- Functionality: AWS Step Functions is a serverless orchestration service that lets you combine AWS Lambda functions and other AWS services to build business-critical applications.
- Why it is used:
  - Orchestration: Manages and visualizes the components of media processing pipelines.
  - Error Handling: Provides built-in error handling, retries, and state management.
  - Justification: Addresses requirements #8a, #8b, and #8c by orchestrating complex workflows, improving manageability, and enhancing error handling.

### g. Amazon API Gateway

- Functionality: Amazon API Gateway is a fully managed service for creating, publishing, maintaining, monitoring, and securing APIs at any scale.
- Why it is used:
  - API Management: Simplifies the process of creating and managing APIs for serverless applications.
  - Security: Integrates with AWS IAM and Amazon Cognito for secure API access.

    - Justification: Supports the serverless architecture by providing managed API creation and management, meeting requirement #4.

### h. Amazon Cognito

- Functionality: Amazon Cognito provides user sign-up, sign-in, and access control, enabling you to authenticate users through social identity providers and enterprise identity providers via SAML 2.0.
- Why it is used:
    - User Management: Simplifies user authentication and authorization.
    - Security: Enhances security by managing user identities and providing secure access to resources.
    - Justification: Enhances security and reduces development effort by offering managed user authentication and authorization services.

### i. AWS IAM (Identity and Access Management)

- Functionality: AWS IAM enables you to manage access to AWS services and resources securely. You can create and manage AWS users and groups and use permissions to allow or deny access to resources.
- Why it is used:
    - Fine-Grained Access Control: Provides detailed permissions management for AWS resources.
    - Security: Enhances the security of the system by ensuring that only authorized users have access to resources.
    - Justification: Improves overall system security by providing fine-grained access control.

### j. Amazon CloudWatch

- Functionality: Amazon CloudWatch is a monitoring and observability service that provides data and actionable insights to monitor your applications, respond to system-wide performance changes, and optimize resource utilization.
- Why it is used:
    - Monitoring: Tracks system performance and health metrics.
    - Alerting: Provides alerting and automated responses to changes in performance.
    - Justification: Ensures the system is monitored and issues are identified and addressed promptly, enhancing system reliability and performance.

**Appendix 2: Requirements fulfillment**

### 1. Use of Managed Cloud Services

- **Requirement:** Where possible the company would like to use managed cloud services to minimize the need for in-house systems administration. Photo and other media will be stored in AWS S3.
- **Fulfillment:**
    - **Amazon S3:** Provides scalable, durable, and secure storage for photos and media, reducing the need for in-house systems administration.
    - **AWS Lambda:** Serverless compute service that minimizes the need for managing servers.
    - **Amazon DynamoDB:** Fully managed NoSQL database service, reducing the need for database administration.
    - **Amazon API Gateway:** Fully managed service for creating and managing APIs, reducing the need for in-house management.
    - **Amazon Cognito:** Managed user authentication and authorization service, reducing the development effort and in-house management.
    - **AWS IAM:** Manages access to AWS resources, reducing the need for in-house access control systems.

### 2. Scalability to Cope with Demand Growth

- **Requirement:** The company expects demand to double every 6 months and wants the architecture to cope with this growth.
- **Fulfillment:**
  - **Amazon S3:** Automatically scales to accommodate growing storage needs.
  - **Amazon CloudFront:** Scales globally to deliver content efficiently.
  - **AWS Lambda:** Scales automatically with the number of requests.
  - **Amazon DynamoDB:** Scales seamlessly to handle increasing data and request rates.
  - **Amazon SQS:** Decouples tasks, allowing the system to handle increased workloads effectively.

### 3. Reduce Compute Capacity Load (EC2 instances not involved)

- **Requirement:** The desired load on compute capacity should decrease to between 50 and 60%.
- **Fulfillment:**
  - **AWS Lambda:** Reduces the need for managing EC2 instances by providing serverless computing.

### 4. Adoption of Serverless/Event-Driven Solution

- **Requirement:** The company would like to adopt a serverless/event-driven solution.
- **Fulfillment:**
  - **AWS Lambda:** Enables a serverless, event-driven architecture.
  - **Amazon SQS:** Facilitates event-driven processing by queuing tasks.
  - **AWS Step Functions:** Orchestrates event-driven workflows.

### 5. Cost-Effective Database Solution

- **Requirement:** The relational database is slow and costly. The company wants a more cost-effective option.
- **Fulfillment:**
  - **Amazon DynamoDB:** Provides a fast, fully managed, and cost-effective NoSQL database solution.

### 6. Improved Global Response Times

- **Requirement:** Global response times need to be improved.
- **Fulfillment:**
  - **Amazon CloudFront:** Enhances global response times by delivering content through a global network of edge locations.

### 7. Handling of Video Media in the Future

- **Requirement:** The system should be extendable to handle video media.
- **Fulfillment:**
  - **Amazon S3:** Can store various types of media including videos.
  - **AWS Lambda:** Can process different media types, including video transcoding.
  - **AWS Step Functions:** Manages workflows that include video processing tasks.

### 8. Automated Media Processing

- **Requirement:** Various versions of media should be automatically produced when uploaded to S3, with an extensible architecture for future processing needs.
- **Fulfillment:**
  - **AWS Lambda:** Triggers processing tasks automatically when media is uploaded to S3.
  - **Amazon SQS:** Queues processing tasks to decouple and manage load.

- **AWS Step Functions:** Orchestrates complex processing workflows.
- **Amazon S3:** Stores both original and processed media.
- **Future Extensibility:** The architecture allows adding AI-based services or other processing tasks in the future.

**Appendix 3: UML Sequence Diagrams**

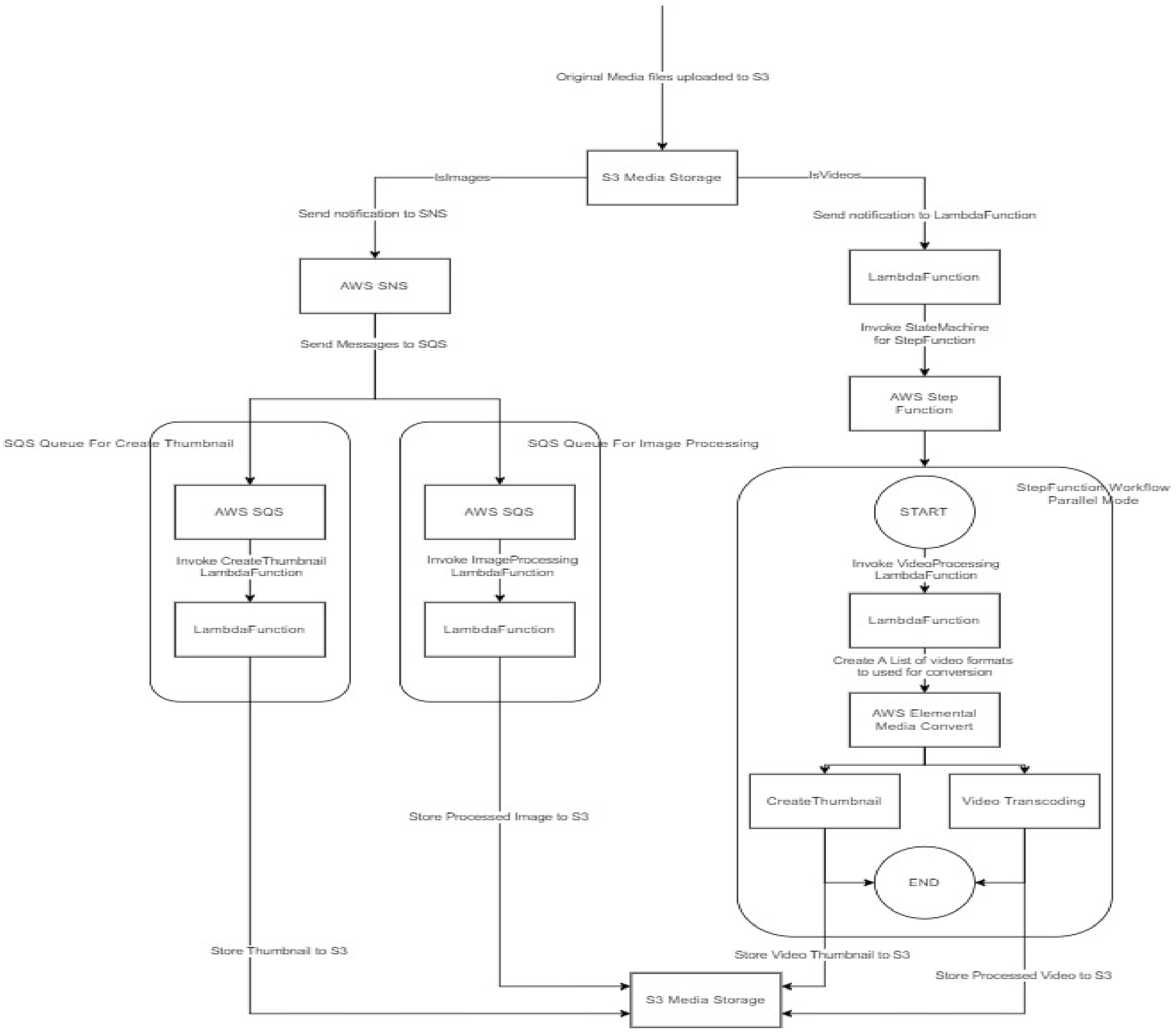
Original Media files uploaded to S3
S3 Media Storage
IsImages
IsVideos
Send notification to SNS
Send notification to LambdaFunction
AWS SNS
LambdaFunction
Send Messages to SQS
Invoke StateMachine for StepFunction
AWS Step Function
SQS Queue For Create Thumbnail
SQS Queue For Image Processing
StepFunction Workflow Parallel Mode
AWS SQS
AWS SQS
START
Invoke CreateThumbnail LambdaFunction
Invoke ImageProcessing LambdaFunction
Invoke VideoProcessing LambdaFunction
LambdaFunction
LambdaFunction
LambdaFunction
Create A List of video formats to used for conversion
AWS Elemental Media Convert
CreateThumbnail
Video Transcoding
Store Processed Image to S3
END
Store Thumbnail to S3
Store Video Thumbnail to S3
Store Processed Video to S3
S3 Media Storage

*Figure 8. Media Processing Workflow using the UML Sequence diagram*

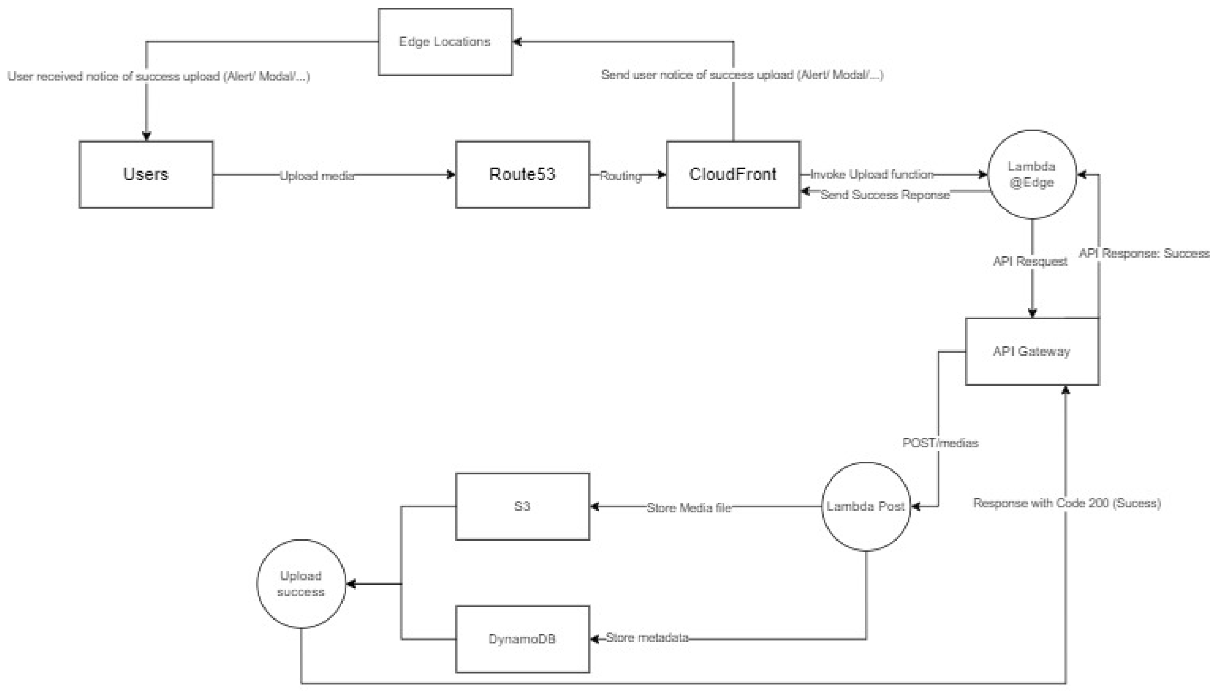

*Figure 9. Users upload media using the UML Sequence diagram*

**Appendix 4: Cost Estimate**

Scenario:

- **Group 1:** 50 customers collectively upload 10 GB of images.
- **Group 2:** 50 customers collectively upload 40 GB of videos.
- **Total of 50GB** by 1000 times upload
- Host is based in Sydney, Australia.
- Picture's size is 150KB.
- Video is in 720p-1080p resolution and 30-60FPS.
- Average upload speed is 21.44Mbps.
- All free trials expired.

| AWS service | Detailed cost | Total |
|---|---|---|
| **1. DynamoDB** | ● **Total images: 68,600 (10 GB / 150 KB per image)**<br>● **Metadata size per image: 100 Bytes**<br>● **Total metadata storage: 68,600 images * 100 Bytes = 6,860,000 Bytes = 6,54 MB (rounded to 20 MB for simplicity)**<br>● **Metadata storage: 20 MB (using 1 GB for simplicity)**<br><br>**Storage Cost:**<br><br>Storage Cost=1 GB×\$0.25=\$0.25<br><br>**Provisioned Throughput Cost:**<br><br>Write Capacity Units (WCUs): Assume 1 WCU | **\$2.75 (per month)** |

| | | |
|---|---|---|
| | =>WCU cost: 1×$1.25=$1.25<br><br>Read Capacity Units (RCUs): Assume 5 RCUs<br><br>=> RCU Cost=5×$0.25=$1.25 | |
| **2. Route 53** | **Hosted Zone (per month in the first 25 hosted zones)**: $0.50<br>**DNS Standard Queries (first 1 billion per month)**: $0.04 | $0.50 + $0.04 = $0.54 |
| **3. CloudFront** | **Free for the first 1TB of data, assuming upload frequency will not exceed this parameter** | $0 |
| **4. S3 Standard for media storage** | **Storage cost ($0.025 per GB)**: $**0.025** x 50= $1.25<br>**PUT, COPY, POST, LIST request ($0,0055 per 1000 request): 1000/1000 * $0.0055 = $0.0055**<br>**Data Transfer IN: free**<br>**Data Transfer OUT (per month)**: $0.114 x 50GB = $5.7 | $6.9555 |
| **5. API gateway** | **API Calls (per million)**:<br>1,000/1,000,000 x $1.29 = $0.00129 | $0.00129 |
| **6. SNS topic** | **HTTP/s ($0.60 per million notifications)**, 1,000 / 1,000,000 x $0.6 = $0.06 | $0.0006 |
| **7.SQS** | **FIFO Queues($0.5 per million requests),** 1,000 / 1,000,000 x $0.5 = $0.05 | $0.0005 |
| **8. Lambda** | **Requests ($0.2 per million requests)**: 1,000 / 1,000,000 x $0.20 = $0.0002<br>**For 10 lambda functions:** $0.002<br><br>**Duration Cost:** $0.00001667 per GB-second (assuming 128MB memory and 100ms execution time): 104,857,600 GB-seconds x 0.1 x $0.00001667=$2.20 | $0.024 |
| **9. CloudWatch** | **Metrics (per month)**: Assuming 8 metrics per Lambda x 10 Lambda functions = 80 metrics.<br>Cost: 80 metrics x $0.30 = $19.2 | $24 |
| **10. AWS IAM** | **Free** | $0 |

| | | |
|---|---|---|
| **11. ELEMENTAL MEDIA CONVERT** | Single-pass (Speed Optimized, $0.0106 per minutes)<br><br>Assuming 4 hours transcoding for 40GB of video: 240 minutes x $0.0106 = $2.544 | $2.544 |
| **Total** | | $36.81589 |

Scenario:

- **Group 1:** 500 customers collectively upload 20 GB of images.
- **Group 2:** 500 customers collectively upload 80 GB of videos.
- **Total of 100GB** by 10000 times upload
- Host is based in Sydney, Australia.
- Picture's size is 150KB.
- Video is in 720p-1080p resolution and 30-60FPS.
- Average upload speed is 21.44Mbps.
- All free trials expired.

| AWS service | Detailed cost | Total |
|---|---|---|
| **1. DynamoDB** | • **Total images: 137,200 (10 GB / 150 KB per image)**<br>• **Metadata size per image: 100 Bytes**<br>• **Total metadata storage: 137,200 images * 100 Bytes =13,720,000 Bytes = 13,08 MB (rounded to 20 MB for simplicity)**<br>• **Metadata storage: 20 MB (using 1 GB for simplicity)**<br><br>**Storage Cost:**<br><br>Storage Cost=1 GB×$0.25=$0.25<br><br>**Provisioned Throughput Cost:**<br><br>Write Capacity Units (WCUs): Assume 1 WCU<br><br>=>WCU cost: 1×$1.25=$1.25<br><br>Read Capacity Units (RCUs): Assume 5 RCUs<br><br>=> RCU Cost=5×$0.25=$1.25 | **$2.75 (per month)** |
| **2. Route 53** | **Hosted Zone (per month in the first 25 hosted zones)**: $0.50 | $0.50 + $0.04 = $0.54 |

| | | |
|---|---|---|
| | **DNS Standard Queries (first 1 billion per month)**: $0.04 | |
| **3. CloudFront** | **Free for the first 1TB of data, assuming upload frequency will not exceed this parameter** | $0 |
| **4. S3 Standard for media storage** | **Storage cost ($0.025 per GB)**: $**0.025** x 100= $2.5<br>**PUT, COPY, POST, LIST request ($0,0055 per 1000 request): 10000/1000 * $0.0055 = $0.055**<br>**Data Transfer IN: free**<br>**Data Transfer OUT (per month)**: $0.114 x 100GB = $11.4 | $13.955 |
| **5. API gateway** | **API Calls (per million)**:<br>10,000/1,000,000 x $1.29 = $0.00129 | $0.0129 |
| **6. SNS topic** | **HTTP/s ($0.60 per million notifications)**, 10,000 / 1,000,000 x $0.6 = $0.06 | $0.006 |
| **7.SQS** | **FIFO Queues($0.5 per million requests),** 10,000 / 1,000,000 x $0.5 = $0.05 | $0.005 |
| **8. Lambda** | **Requests ($0.2 per million requests)**: 10,000 / 1,000,000 x $0.20 = $0.02<br>**For 10 lambda functions:** $0.2<br><br>**Duration Cost:** $0.00001667 per GB-second (128MB on 1ms):<br><br>10.48575M x 0.128GB x 0.1s x $0.00001667 = $2.2 | $2.4 |
| **9. CloudWatch** | **Metrics (per month)**: Assuming 10 metrics per Lambda x 8 Lambda functions = 64 metrics.<br>Cost: 80 metrics x $0.30 = $19.2 | $24 |
| **10. AWS IAM** | **Free** | $0 |
| **11. ELEMENTAL MEDIA CONVERT** | Single-pass (Speed Optimized, $0.0106 per minutes)<br><br>Assuming 8 hours transcoding for 80GB of video: 480 minutes x $0.0106 = $5,88 | $5,88 |
| **Total** | | $49.5489 |

## 7. References

[1] Amazon Web Services, "Amazon Route 53 Pricing," Amazon Web Services, Inc. [Online]. Available: Amazon Route 53 pricing - Amazon Web Services. [Accessed: Aug. 2, 2024].
[2] Amazon Web Services, "AWS Identity and Access Management," Amazon Web Services, Inc. [Online]. Available: Access Management- AWS Identity and Access Management (IAM) - AWS (amazon.com). [Accessed: Aug. 2, 2024].
[3] Amazon Web Services, "Pricing for On-Demand Capacity," Amazon Web Services, Inc. [Online]. Available: Amazon DynamoDB Pricing for On-Demand Capacity. [Accessed: Aug. 2, 2024].
[4] Amazon Web Services, "Amazon CloudFront Pricing," Amazon Web Services, Inc. [Online]. Available: Amazon CloudFront CDN - Plans & Pricing - Try For Free. [Accessed: Aug. 2, 2024].
[5] Amazon Web Services, "Amazon S3 Pricing," Amazon Web Services, Inc. [Online]. Available: Giá Amazon S3 Simple Storage Service - Amazon Web Services. [Accessed: Aug. 2, 2024].
[6] Amazon Web Services, "Amazon API Gateway Pricing," Amazon Web Services, Inc. [Online]. Available: Amazon API Gateway Pricing | API Management | Amazon Web Services. [Accessed: Aug. 2, 2024].
[7] Amazon Web Services, "Amazon SNS Pricing," Amazon Web Services, Inc. [Online]. Available: Amazon Simple Notification Service (SNS) Pricing | Messaging Service | AWS. [Accessed: Aug. 2, 2024].
[8] Amazon Web Services, "Amazon SQS Pricing," Amazon Web Services, Inc. [Online]. Available: Amazon SQS Pricing | Message Queuing Service | AWS. [Accessed: Aug. 2, 2024].
[9] Amazon Web Services, "Amazon Lambda Pricing," Amazon Web Services, Inc. [Online]. Available: Serverless Computing – AWS Lambda Pricing – Amazon Web Services. [Accessed: Aug. 2, 2024].
[10] Amazon Web Services, "Amazon CloudWatch Pricing," Amazon Web Services, Inc. [Online]. Available: Amazon CloudWatch Pricing – Amazon Web Services (AWS). [Accessed: Aug. 2, 2024].
[11] Amazon Web Services, "Amazon Rekognition Pricing," Amazon Web Services, Inc. [Online]. Available: Amazon Rekognition – Pricing - AWS. [Accessed: Aug. 2, 2024].
[12] Amazon Web Services, "Amazon Elemental MediaConvert Pricing," Amazon Web Services, Inc. [Online]. Available: MediaConvert Pricing (amazon.com) [Accessed: Aug. 2, 2024].
[13] Amazon Web Services, "AWS Lambda Features," Amazon Web Services, Inc., [Online]. Available: Lambda Features - AWS [Accessed: Aug. 2, 2024].
[14] Amazon Web Services, "Amazon DynamoDB Features," Amazon Web Services, Inc., [Online]. Available: DynamoDB Features - AWS [Accessed: Aug. 2, 2024].
[15] Amazon Web Services, "Amazon S3 Features," Amazon Web Services, Inc., [Online]. Available: S3 Features - AWS [Accessed: Aug. 2, 2024].
[16] Amazon Web Services, "Amazon CloudFront Features," Amazon Web Services, Inc., [Online]. Available: CloudFront Features - AWS [Accessed: Aug. 2, 2024].
[17] Amazon Web Services, "Serverless Workflows with AWS Step Functions," Amazon Web Services, Inc., [Online]. Available: Serverless workflows with Step Functions - AWS [Accessed: Aug. 2, 2024].
[18] Amazon Web Services, "AWS Elemental MediaConvert Features," Amazon Web Services, Inc., [Online]. Available: MediaConvert Features - AWS [Accessed: Aug. 2, 2024].
[19] Amazon Web Services, "Amazon SQS Features," Amazon Web Services, Inc., [Online]. Available: SQS Features - AWS [Accessed: Aug. 2, 2024].
[20] Amazon Web Services, "AWS Identity and Access Management (IAM)," Amazon Web Services, Inc., [Online]. Available: IAM - AWS [Accessed: Aug. 2, 2024].
[21] Amazon Web Services, "Amazon API Gateway Features," Amazon Web Services, Inc., [Online]. Available: API Gateway Features - AWS [Accessed: Aug. 2, 2024].