COS20019 – Cloud Computing Architecture Assignment 3

Serverless/Event-driven Architecture Design Report

Hai Nam Ngo - 103488515 Kelvin Le - 104057343 Moshfeque E Jahan - 103803710

I. ABSTRACT (Hai Nam)

With the evolution of the digital age, cloud-based platforms, especially Amazon Web Services (AWS), have become crucial in shaping online infrastructure. AWS facilitates users by overseeing the foundational elements, empowering developers to concentrate primarily on application functionality, rather than intricate technicalities such as server management and network configurations. This document presents a blueprint and execution strategy for a serverless/event-driven framework aiming to optimize business operations. AWS Lambda, API Gateway, DynamoDB, SQS, and more will be explored in-depth. Besides proposing the architectural framework, the study delves into the logic behind the choices made. Lastly, a glance at potential alternatives to the proposed architecture is provided.

II. INTRODUCTION (Hai Nam)

The integration of digital innovations into business processes has become pivotal in recent times. The advent of cloud computing has ushered in a novel era, offering enterprises and software developers new avenues to consolidate their online presence. This paradigm shift in cloud-based solutions offers both financial viability and efficiency in infrastructure management. The transition from on-site to cloud-based web operations has been streamlined, enhancing the web user experience. Significantly, AWS's self-reliant offerings have fostered the emergence of serverless and event-driven frameworks, which augments web operational agility and efficacy. This study advocates for the implementation of a serverless/event-driven structure utilizing AWS tools tailored to a business's unique operational conditions. A comprehensive analysis of the company's operations is imperative before outlining the design.

III. BUSINESS SCENARIO (Hai Nam)

The Photo Album platform has witnessed unprecedented growth, necessitating further enhancements to fulfill burgeoning user demands. Specific challenges and prerequisites have been pinpointed, and strategies need to be devised to address them, ensuring an unparalleled user experience.

- Managed Cloud Services and Media Storage:
  - Use AWS's managed services for most of our infrastructure to reduce the need for in-house system administration.
  - Store photos and other media in AWS S3 buckets, such as the Web Store in Bucket, Original Media Bucket, and Transcoded Media Bucket, ensuring durability, availability, and scalable object storage

- Scalability:
  - Leverage AWS Auto Scaling, represented in the diagram as AWS Lambda icons, which adjust based on the triggers.
  - Adopt Amazon API Gateway for scalable and secure API deployment. It interfaces with services like AWS Amplify, which auto-scales based on traffic.

- Serverless/Event-Driven Architecture:
  - Utilize AWS Lambda, which gets triggered by events such as media uploads to S3 or tasks in the SQS queue.
  - Use Amazon DynamoDB to store metadata, as indicated in the diagram.

- Global Performance:
  - Implement a CDN cache, likely using Amazon CloudFront, to cache content in edge locations around the world, ensuring low-latency access.

- Media Processing and Transformation:
  - Use AWS Step Functions to manage media transformation workflows. When media is uploaded, the media files get processed by workflows like the ones specified for images and videos in the diagram.

- Platform Suitability:
  - Distribute the workload based on the intensity and nature of the task. AWS Lambda handles short-lived tasks, while more intense processing could be managed by dedicated resources.

- Decoupling and load management:
  - Use Amazon SQS to decouple application components and manage tasks in a queue. This ensures that the application can handle varying loads, as seen with SQS integration in the media processing workflows.

- Futureproofing and Extensions:
  - AWS recognition can be integrated later for automatic image and video analysis.
  - AWS Elemental Media Convert, although not shown in the diagram, can be integrated in the future for video transcoding.

IV. ARCHITECTURE DESIGN (Hai Nam)

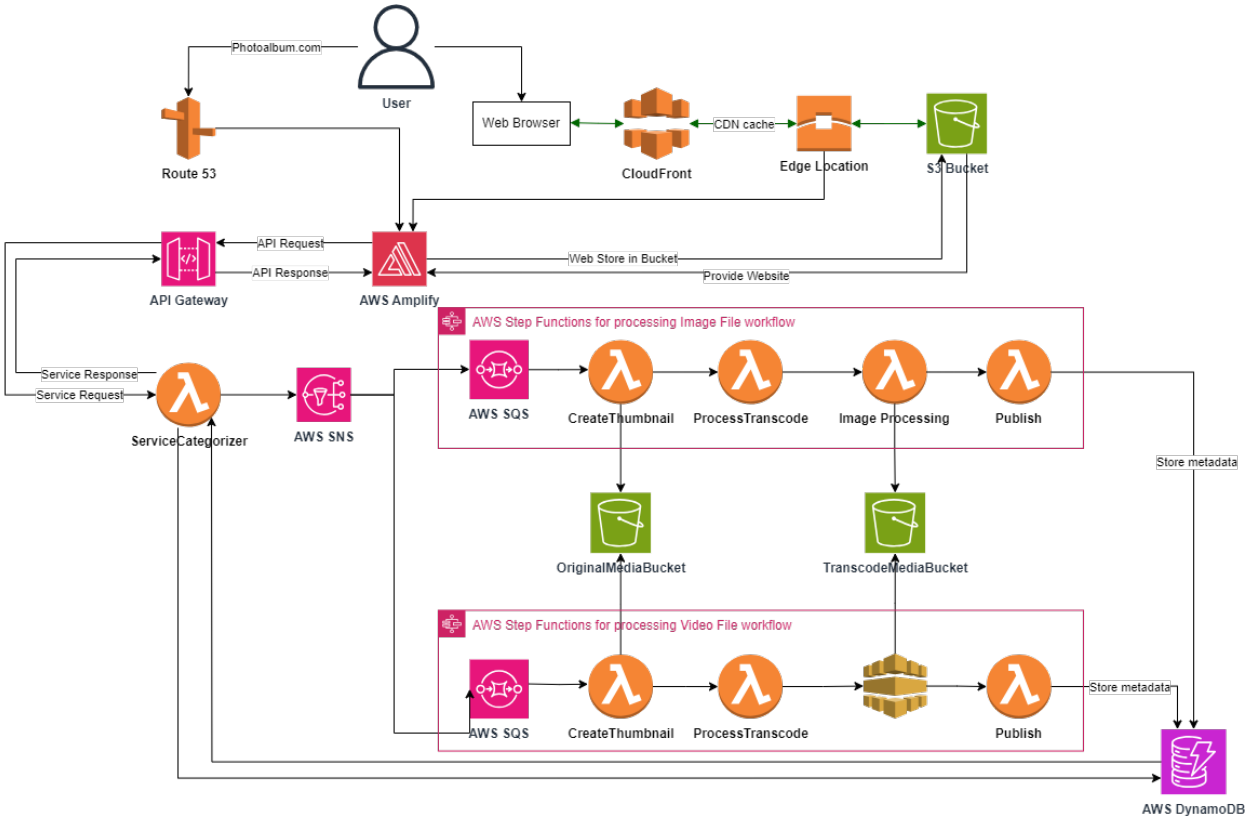

Figure 1 AWS Architecture Diagram

A. AWS Content Delivery Mechanism (Hai Nam)

With the rapid surge in our website's user base, it's imperative to implement a robust content delivery mechanism that not only enhances the user experience but also provides us with a competitive advantage. By leveraging AWS solutions like CloudFront, we can ensure seamless content delivery.

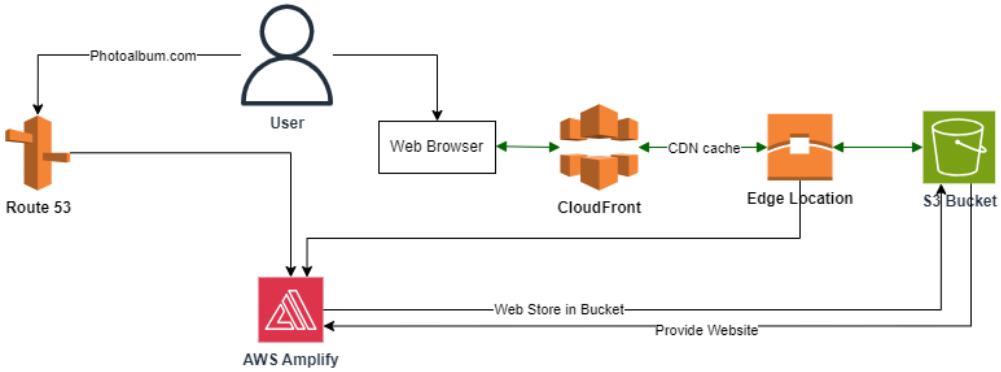

Figure 2 AWS Content Delivery Mechanism

Users can access our platform in two main ways: Firstly, they can enter our website through a domain, which is then resolved by a service analogous to AWS Route 53. This resolution directs them to our site hosted on AWS Amplify. According to research [1], utilizing a reliable domain resolution system like Route 53 can greatly bolster our online presence, offering a memorable digital identity.

Furthermore, our platform has a media delivery infrastructure reminiscent of cloud storage platforms like Google Drive. This is where AWS CloudFront steps in. It delivers media content to users from the nearest edge location for optimal speed. If the required content isn't in the CloudFront cache, the system fetches it from our primary storage, ensuring that users always have a swift and uninterrupted experience.

B. AWS Serverless Three-Tier Architecture (Hai Nam)

A popular rendition of the multi-layered model [2] is the tri-layered approach, incorporating three core layers: the user interface layer, the processing layer, and the storage layer.

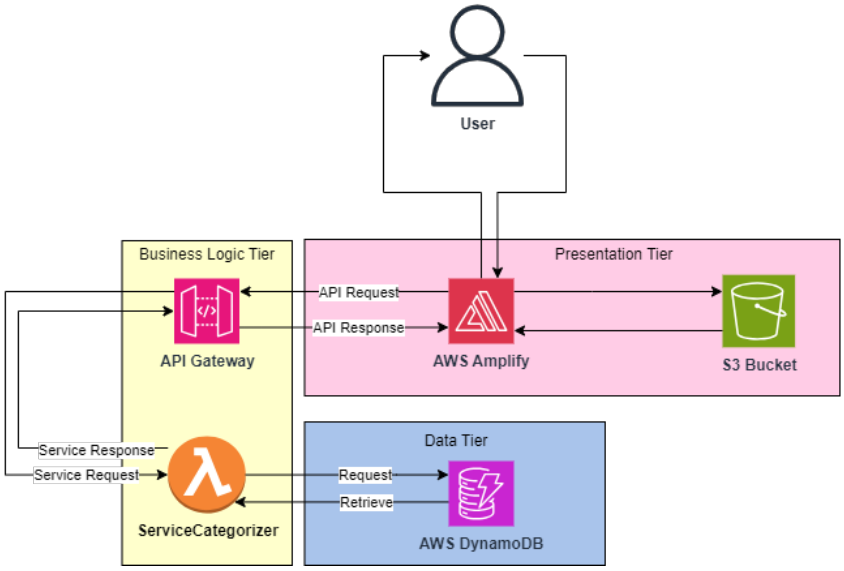

Figure 3 AWS Serverless Three-Tier Architecture

Static web elements are anchored in Amazon S3 and showcased through Amazon Amplify, as shown in the architecture, representing our user interface layer. According to research [2,] this is where API Gateway and AWS Lambda can outperform a traditional, server-based implementation. Finally, for the storage layer, AWS DynamoDB serves as our long-term data vault.

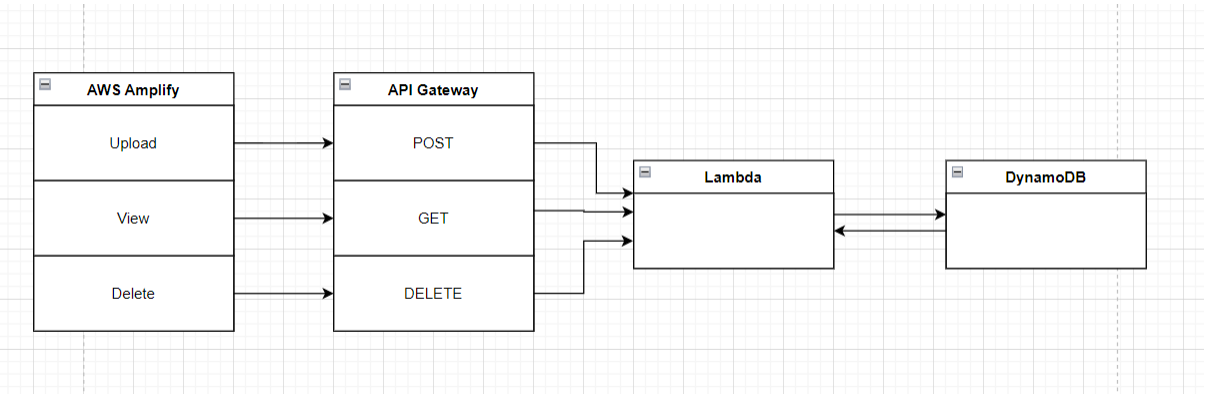

Figure 4 Web architecture UML Diagram

C. Dependable Data Dissemination Model (Hai Nam)

The data dissemination model is a renowned AWS blueprint, ensuring the adept relay of information to manifold subsequent services or systems.

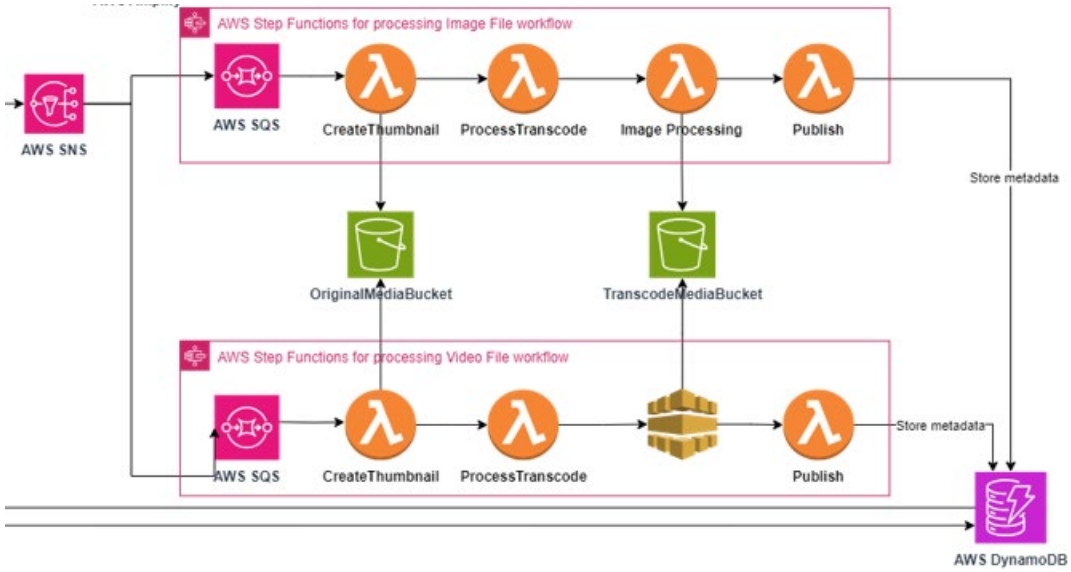

Figure 5 Dependable Data Dissemination Model

D. Multimedia Processing Sequence (Hai Nam)

In this workflow, an SQS queue triggers a step function. The initial action is undertaken by an AWS Lambda function that generates a thumbnail of the uploaded content, saving it alongside the original in an S3 folder designated for original media. Subsequently, a separate Lambda function readies the media for transcoding, which includes determining the website's supported formats and creating metadata for the soon-to-be-transcoded file. Elastic Transcoder then converts the media to the specified format. After this, the media is passed to the publishing Lambda function. The process culminates with the transcoded media being saved in the “TranscodedMedia” S3 folder, and its metadata stored in DynamoDB. This entire mechanism is both adaptable and repeatable, facilitating the easy inclusion or alteration of processing steps as demands shift. For instance, if we wish to incorporate AI detection into our workflow, we can introduce an additional Lambda function at the necessary juncture using AWS Rekognition.

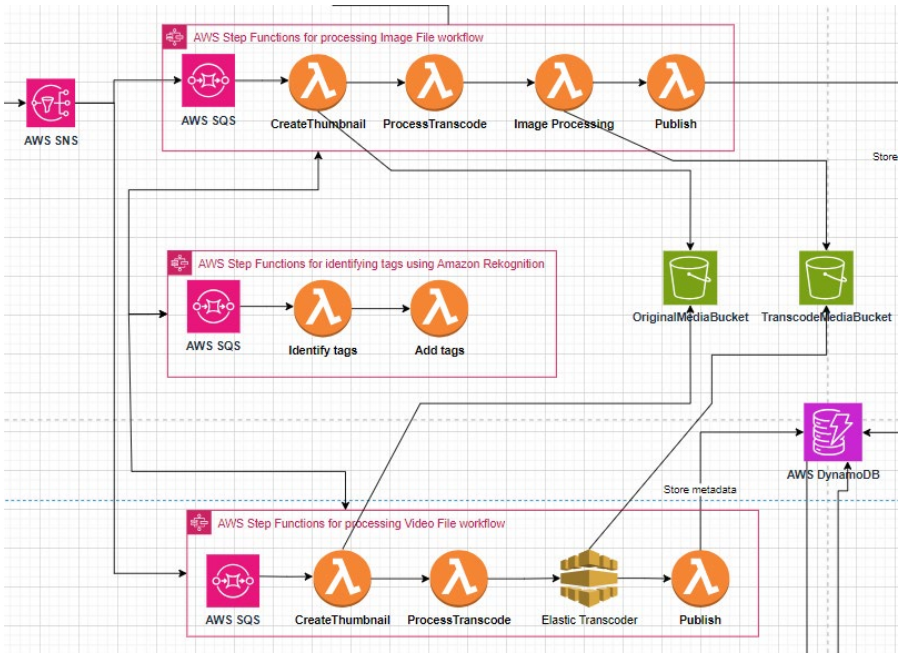

Figure 6 Media Processing Step

E. UML Diagram (Mosh)

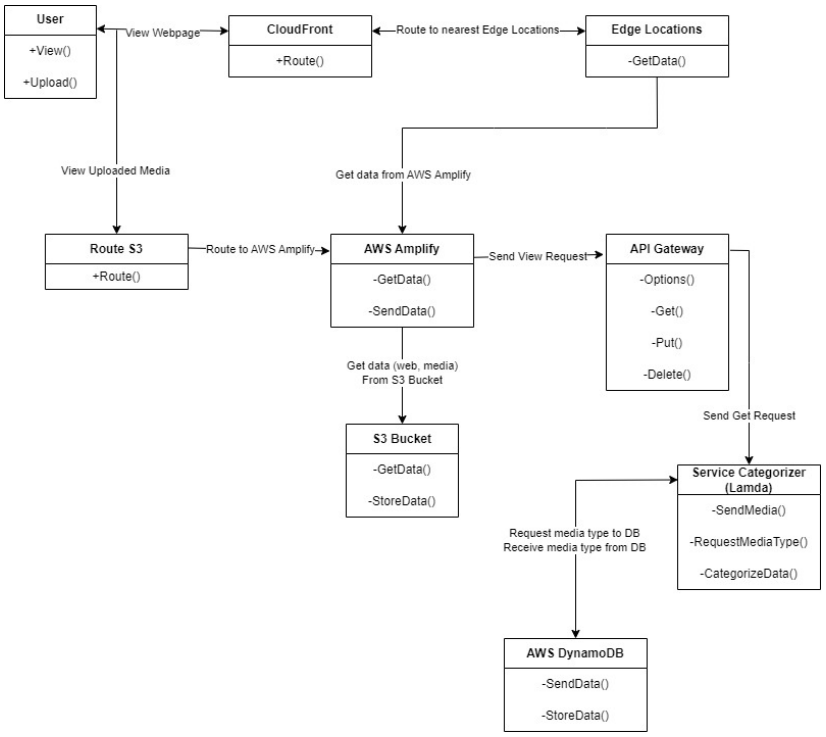

Figure 7 Viewing UML Diagram

When users access the website, they will be directed to the closest edge location using CloudFront, and AWS Amplify will then showcase the website and certain media files stored in the bucket. However, the process varies when it comes to viewing uploaded media. In this case, the user's request is initially routed through CloudFront via Route

53. AWS Amplify sends a request to obtain specific media metadata to the API Gateway, which, in turn, queries the database for the requested metadata with assistance from the Categorizer function.

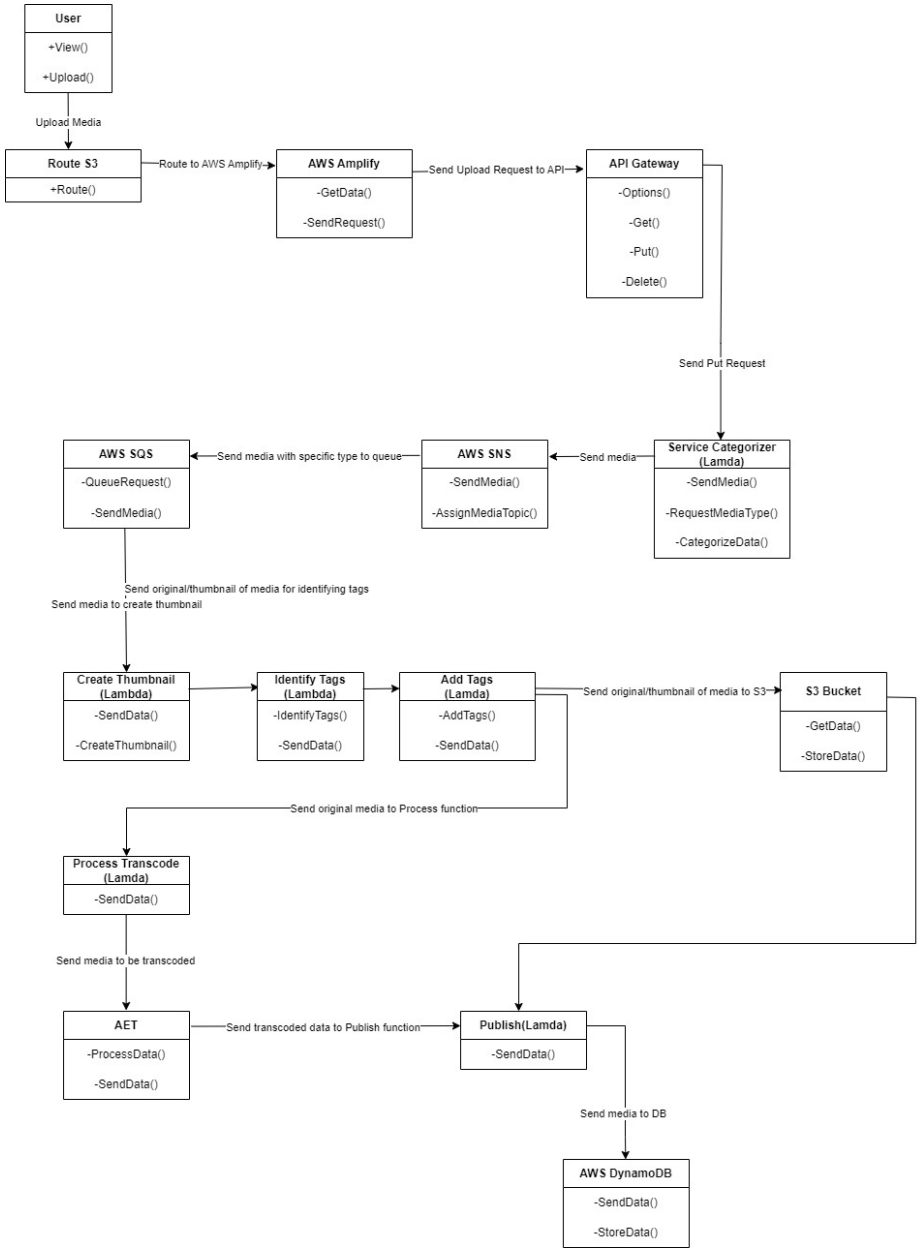

Figure 8 Uploading UML Diagram

When a user initiates the upload of media, the process begins with Route 53 and CloudFront, which directs the media to AWS Amplify. Subsequently, AWS Amplify generates an upload request through an API, triggering the PUT method. The media is then routed through the API and directed to the Service Categorizer function, which assigns a specific category based on its media type, such as image topic or video topic. These categories inform AWS SNS about the intended destination for the media. AWS SNS then dispatches the media to AWS SQS for queuing and processing.

At this stage, both the original media and its thumbnail has tags added using AWS Rekognition and is then stored in an S3 bucket, thanks to the creation of the thumbnail. Simultaneously, the media undergoes transcoding

using AET, resulting in the generation of metadata. The transcoded media is placed into S3 once again, while the associated metadata is securely stored within the DynamoDB database.

V. DESIGN RATIONALE (Mosh)

With a thorough discussion of our AWS architecture in place, it's time to delve into how this design addresses the identified pain points in the given business scenario. Additionally, we'll explore an alternative solution and the design criteria that guided our choices

A. Viewing and Uploading media (Mosh)

In the original architecture, users seeking to view media content are routed through CloudFront to an EC2 instance hosting the website. This website resides within a private subnet, accessible to users via a NAT gateway.

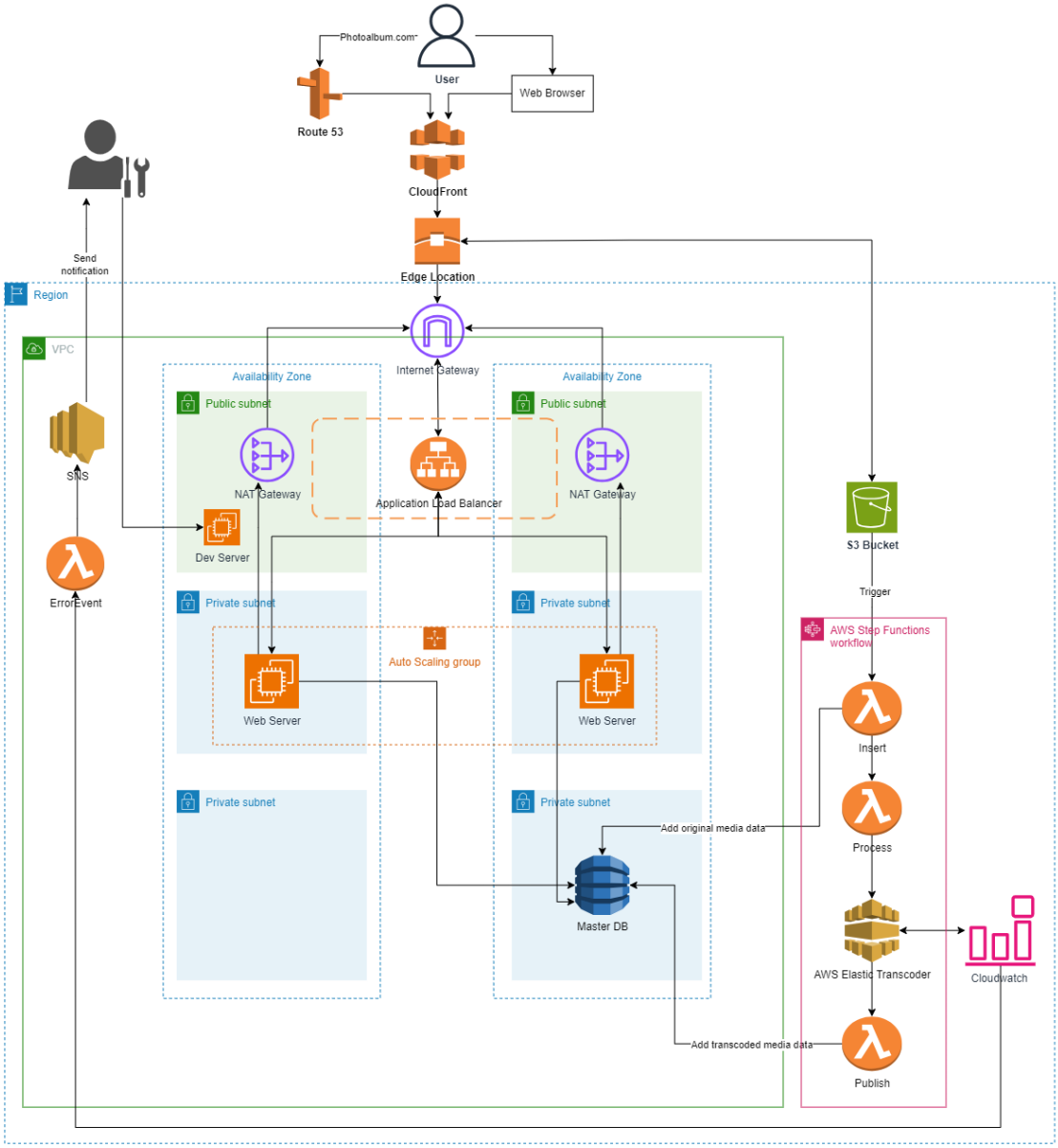

Figure 9 Original architecture (Initial Plan)

However, the alternative architecture simplifies this process. In the second design, users access the website with its source code stored in an S3 Bucket via AWS Amplify. This approach is notably more straightforward, as it eliminates the need for instances and gateways, enhancing the architecture's high availability.

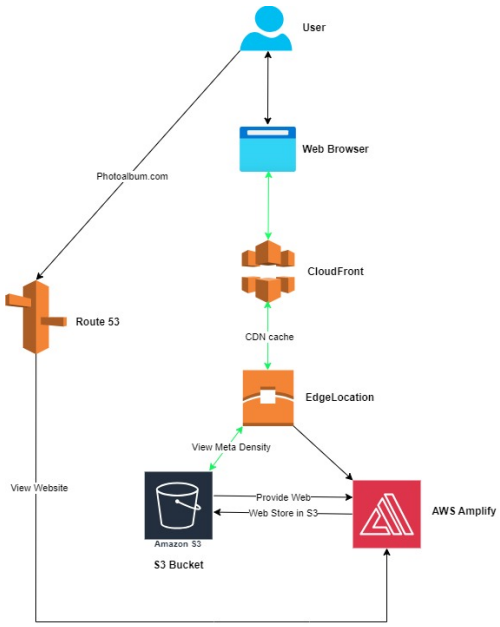

Figure 10 Using AWS Amplify for the website

Regarding media upload, the original design directs media to the AWS Rekognition first to identify and add tags then it is directed to the S3 bucket, which triggers a step function to process and return the media to the bucket. Additionally, metadata is sent to the database, as is the case for metadata processed by AWS Transcoder.

The alternative approach to media upload is distinct. As media passes through AWS Amplify, it triggers an API to send a PUT request to the ServiceCategorizer, which subsequently forwards the media to the Reliable Fanout process responsible for handling media types.

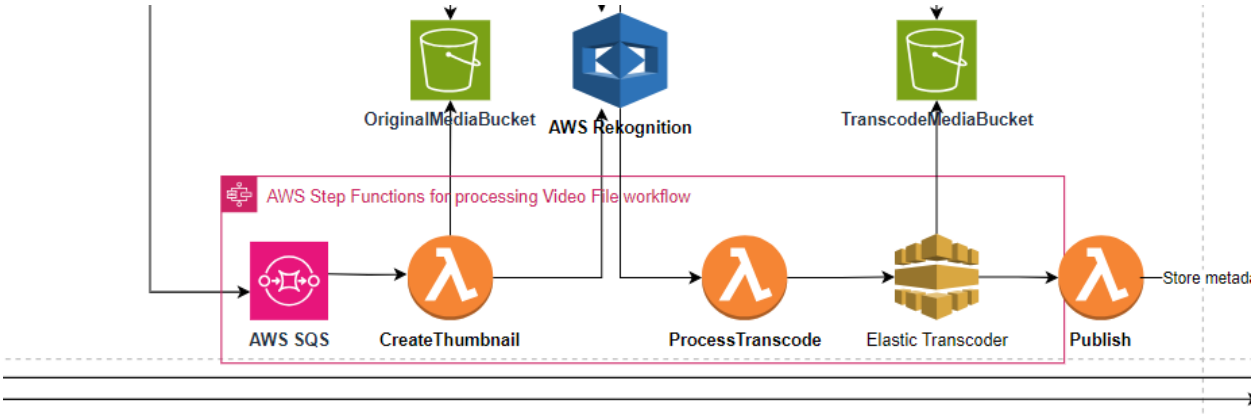

Figure 11 Media Uploading Process

A. VPC vs. Serverless three-tier architecture (Mosh)

The serverless three-tier architecture was chosen for its ease of management. AWS automates critical aspects such as scaling and security. In contrast, the original VPC-based architecture requires manual monitoring and updates, even with an auto-scaling group in place. The serverless approach, utilizing AWS Amplify, significantly reduces administrative overhead [3], allowing administrators to focus on performance enhancement.

Security is another critical consideration. In the VPC environment, administrators must manually configure each component as a security group or NACL, which can be a cumbersome and challenging task. In contrast, the architecture in use leverages AWS Amplify for security, providing authentication, authorization, data encryption, and further bolstering security [4] with IAM roles.

It's worth noting that the serverless three-tier architecture may have certain infrastructure limitations, but these are of secondary concern given the substantial benefits it offers.

In summary, the alternative serverless architecture outperforms the original VPC-based design in terms of management, security, and operational simplicity, making it the preferred choice for our application's needs.

B. Design Criteria (Mosh)

Upon thorough evaluation of the AWS services, the subsequent sections will elaborate on the design standards for each facet of the well-architected framework.

1) Performance

a) Criteria:

- Ensuring swift media uploads and downloads
- Streamlining storage and retrieval of images
- Facilitating rapid website loading with minimal latency, even under high traffic conditions and for users located outside of Australia

b) Solution:

- Leveraging CloudFront for caching static website content and expedited content delivery through edge locations
- Employing Route 53 to intelligently route user traffic to the nearest CloudFront edge location based on their geographical proximity
- Enhancing long-distance file transfers with S3 Transfer Acceleration, a feature embedded within S3, resulting in accelerated content transfers by up to 50-500%
- Implementing CloudWatch to proactively monitor and assess website performance metrics, including latency, error rates, and resource utilization
- Utilizing Elastic Transcoder for media optimization, enabling the seamless conversion of media into appropriate formats and resolutions tailored for web viewing.

2) Scalability

a) Criteria:

- Ensuring the website's capacity to manage burgeoning traffic volumes and escalating storage demands
- Efficiently distributing workloads across multiple servers or regions
- Dynamically adjusting resources in response to traffic patterns
- Establishing a highly reusable and adaptable system architecture to facilitate future modifications, such as the integration of AI-based features like media tag identification

b) Solution:

- Utilizing CloudFront to expand the global content delivery network, adeptly handling traffic spikes
- Creating a scalable RESTful API for the website's backend using API Gateway, equipped to scale as necessitated by incoming API requests
- Employing AWS Amplify to effortlessly scale both the frontend and backend components of the application
- Implementing DynamoDB for the scalable storage and retrieval of metadata
- Integrating AWS Rekognition, which uses AI to identify and add tags to the media
- Leveraging S3 for the expansion of content storage and retrieval capabilities
- Harnessing Amazon SNS and SQS for efficient messaging and queuing mechanisms, capable of managing substantial request volumes

- Designing system components with a decoupled architecture, ensuring flexibility for future modifications, such as the seamless addition of new Lambda functions to introduce novel features.

3) Reliability

a) **Criteria**:

- Ensuring the website's high availability and swift recovery from potential failures

- Implementing regular backups as a preventive measure against data loss

b) **Solution**:

- Adopt Amazon S3 for resilient and readily available storage solutions.
- Utilize DynamoDB to replicate data across multiple regions, thus guaranteeing a high level of availability.
- Leverage Amazon DynamoDB for a scalable and highly available NoSQL database system.
- Employ CloudFront to effectively manage failovers and deliver content from the closest accessible edge location, bolstering reliability.

4) Security

a) **Criteria**:

- Ensure the safeguarding of data, encompassing personal information, uploaded media, and other sensitive content.
- Establish comprehensive protective measures to thwart unauthorized access, data theft, or any potential data destruction within the website.

b) **Solution**:

- Implement AWS Identity and Access Management (IAM) for the meticulous control of access and permissions.
- Employ Amazon S3 as a secure object storage solution, fortified with granular access control.
- Utilize CloudWatch for the continuous monitoring and early detection of security threats and anomalies.
- Enforce authentication and authorization mechanisms in the backend API through API Gateway to fortify the website's protection against unauthorized access.

5) Cost estimation (Hai Nam)

According to research [6], for cost optimization, there are trade-offs to consider. For example, whether to prioritize speed to market or cost. In some cases, it's better to prioritize speed—getting to market quickly, shipping new features on time, or meeting a deadline, rather than investing in upfront cost reduction.

Based on the data given, the application's demand is doubling every six months and is projected to continue this growth pattern for another 2 to 3 years. We assume that there will be 30,000 users with a usage of 10TB for a month. And every 6 months, the number of users and the demanding memory will be doubled.

We will use AWS Pricing Calculator to calculate the exact cost for 64GB, and then double each time for larger GB. The total amount of cost will cover Route 53, API Gateway, AWS amplify, CloudFront, S3 Bucket, Lambda function, SNS, SQS, DynamoDB and Elastic transcoder.

Estimate summary

Upfront cost	Monthly cost	Total 12 months cost
6,750.00 USD	50,721.33 USD	615,405.96 USD Includes upfront cost

Detailed Estimate

Name	Group	Region	Upfront cost	Monthly cost
Amazon Route 53	No group applied	US East (N. Virginia)	0.00 USD	634.80 USD
Status: **Description:** **Config summary:** IP (CIDR) blocks (1000), Additional Records in Hosted Zones (10000), Hosted Zones (3), Number of Elastic Network Interfaces (2), Number of domains stored (30000), Basic Checks Within AWS ()				
Amazon API Gateway	No group applied	US East (N. Virginia)	0.00 USD	88.42 USD
Status: - **Description:** **Config summary:** REST API request units (millions), Cache memory size (GB) (None), WebSocket message units (millions), HTTP API requests units (millions), Average size of each request (200 KB), Average message size (128 KB), Requests (10 per month), Requests (10 per month), Messages (10 per month), Average connection duration (15 minutes), Average connection rate (30000 per day)				
AWS Amplify	No group applied	US East (N. Virginia)	0.00 USD	4,389.14 USD
Status: - **Description:** **Config summary:** Duration of each request (in ms) (500), Number of build minutes (259200 per month), Data stored per month (10 TB), Data served per month (10 TB), Number of SSR requests (30000 per day)				
Amazon CloudFront	No group applied	US East (N. Virginia)	0.00 USD	1,805.23 USD
Status: - **Description:** **Config summary:** Data transfer out to internet (1 TB per month), Data transfer out to origin (1 TB per month), Number of requests (HTTPS) (5000000 per month), Data transfer out to internet (1 TB per month), Data transfer out to origin (1 TB per month), Number of requests (HTTPS) (5000000 per month), Data transfer out to internet (1 TB per month), Data transfer out to internet (1 TB per month), Data transfer out to origin (1 TB per month), Number of requests (HTTPS) (5000000 per month), Number of requests (HTTPS) (5000000 per month), Data transfer out to internet (1 TB per month), Data transfer out to origin (1 TB per month), Number of requests (HTTPS) (5000000 per month), Number of requests (HTTPS) (5000000 per month), Data transfer out to internet (1 TB per month), Data transfer out to origin (1 TB per month), Number of requests (HTTPS) (5000000 per month), Data transfer out to internet (1 TB per month), Data transfer out to origin (1 TB per month), Number of requests (HTTPS) (5000000 per month), Data transfer out to internet (1 TB per month), Data transfer out to origin (1 TB per month), Number of requests (HTTPS) (5000000 per month), Data transfer out to internet (1 TB per month), Data transfer out to origin (1 TB per month), Number of requests (HTTPS) (5000000 per month)				
Amazon Simple Storage Service (S3)	No group applied	US East (N. Virginia)	0.00 USD	1,189.63 USD
Amazon Simple Storage Service (S3)	No group applied	US East (N. Virginia)	0.00 USD	1,189.63 USD
Status: - **Description:** **Config summary:** Data returned by S3 Select (10 TB per month), Data scanned by S3 Select (10 TB per month), PUT, COPY, POST, LIST requests to S3 Standard (900000), GET, SELECT, and all other requests from S3 Standard (900000), S3 Standard storage (10 TB per month) DT Inbound: Internet (10 TB per month), DT Outbound: Internet (10 TB per month)				
AWS Lambda	No group applied	US East (N. Virginia)	0.00 USD	22.23 USD
Status: - **Description:** **Config summary:** Invoke Mode (Buffered), Number of requests (900000 per month), Architecture (x86), Architecture (x86), Time for which Provisioned Concurrency is enabled (2628000 seconds), Number of requests (900000 per month), Number of requests for Provisioned Concurrency (2628000 per month), Amount of ephemeral storage allocated (512 MB)				
Amazon Rekognition	No group applied	US East (N. Virginia)	0.00 USD	31,900.00 USD
Status: - **Description:** **Config summary:** Number of images processed with labels API calls per month (9000000 per month), Number of images processed with content moderation API calls per month (9000000 per month), Number of images processed with detect text API calls per month (9000000 per month), Number of images processed with celebrity API calls per month (9000000 per month), Number of images processed with PPE detection API calls per month (9000000 per month), Number of images processed with Image Properties API calls per month (9000000 per month), Number of DetectFaces API calls per month (9000000 per month), Number of CompareFaces API calls per month (0 per month), Number of IndexFaces API calls per month (0 per month), Number of AssociateFaces API calls per month (0 per month), Number of SearchUsers API calls per month (0 per month), Number of SearchUsersByImage API calls per month (0 per month), Number of SearchFaces API calls per month (0 per month)				
Amazon Elastic Transcoder	No group applied	US East (N. Virginia)	0.00 USD	0.00 USD
Status: - **Description:** **Config summary:** Video assets to transcode (900000 per month), SD output formats count ()				
Amazon DynamoDB	No group applied	US East (N. Virginia)	6,750.00 USD	7,352.63 USD
Status: - **Description:** **Config summary:** Table class (Standard), Average item size (all attributes) (200 KB), Write reserved capacity term (1 year), Read reserved capacity term (1 year), Data storage size (10 TB)				
Amazon Simple Queue Service (SQS)	No group applied	US East (N. Virginia)	0.00 USD	1,010.70 USD
Status: - **Description:** **Config summary:** DT Inbound: Internet (10 TB per month), DT Outbound: Internet (10 TB per month), Standard queue requests (100 million per month), FIFO queue requests (100 million per month), Data transfer cost (921.6)				
Amazon Simple Notification Service (SNS)	No group applied	US East (N. Virginia)	0.00 USD	1,138.92 USD

Figure 12 AWS Pricing Calculation for 10TB and 30,000 users

Cost estimation for 3 years

Year	Month (Duration)	Amount of users	Number of TB needed	Cost for 1 month	Cost for 6 months	
1	6 months	30000.00	10	$ 50,721.33	$ 304,327.98	
1	6 months	60000.00	20	$ 101,442.66	$ 608,655.96	
2	6 months	120000.00	40	$ 202,885.32	$ 1,217,311.92	
2	6 months	240000.00	80	$ 405,770.64	$ 2,434,623.84	
3	6 months	480000.00	160	$ 811,541.28	$ 4,869,247.68	
3	6 months	960000.00	320	$ 1,623,082.56	$ 9,738,495.36	
				Total	**$ 19,172,662.74**	**USD**
					$ 30,292,807.13	**AUD**
				Average cost(per month)	**$ 841,466.86**	**AUD**

Figure 13 Cost estimation for 2 - 3 years

VI. Fulfillment of the business scenario requirements (Kelvin)

In this section, we will outline how our serverless/event-driven architecture addresses the specific requirements of the Photo Album business scenario and provide justification for the architectural decisions made.

1) Managed Cloud Services and Media Storage

Our business scenario outlined the need for managed cloud services and robust media storage capabilities. To fulfill these requirements, we have leveraged AWS's managed services. AWS S3 buckets, such as the Web Store in Bucket, Original Media Bucket, and Transcoded Media Bucket, are utilized for storing photos and media. AWS's managed services handle various aspects of infrastructure management, reducing the operational overhead for the Photo Album platform. Additionally, storing media in AWS S3 buckets ensures durability, availability, and scalable object storage, meeting the platform's media storage requirements.

2) Scalability

Scalability was a key consideration in response to the growing user base in our business scenario. Our architecture incorporates several mechanisms to address this crucial requirement. Utilizing AWS Auto Scaling for AWS Lambda, the system can dynamically adjust its resources as the number of events, such as media uploads or tasks in the SQS queue, increases, ensuring it can effortlessly meet the rising demand. Additionally, Amazon API Gateway plays a pivotal role in providing both scalability and security for API deployment. Integrating SQS into the media processing workflows enhances load management efficiency. This architectural approach seamlessly aligns with the demands of our business scenario, focusing on managing increased traffic and workloads effectively.

In practice, our architecture leverages AWS Auto Scaling and Amazon API Gateway to guarantee scalability. AWS Lambda icons symbolize the use of AWS Auto Scaling, which intelligently adapts resources in response to traffic fluctuations. Meanwhile, Amazon API Gateway interfaces seamlessly with services like AWS Amplify, further enhancing the platform's capacity to accommodate a growing user base and varying workloads, thus fully satisfying the scalability requirements.

3) Serverless/Event-Driven Architecture

The business scenario recommended a serverless/event-driven architecture to enhance operational agility and efficacy. We've adhered to this recommendation by heavily utilizing AWS Lambda, which is triggered by events like media uploads to S3 or tasks in the SQS queue. This serverless design minimizes the need for manual server management and aligns with the business's goals to optimize business operations. AWS Lambda, DynamoDB, and other serverless components are perfectly aligned with the serverless/event-driven architecture requirements.

AWS Lambda functions handle various tasks and processes, getting triggered by events such as media uploads to S3 or tasks in the SQS queue. This approach ensures that the platform can efficiently respond to changes and user interactions without the need for traditional server management.

Amazon DynamoDB is used to store metadata, providing a flexible and scalable data storage solution [5] for the platform's needs. Additionally, Amazon SQS helps decouple application components and manage tasks in a queue, ensuring the platform can handle varying workloads effectively. This approach not only fulfils the business scenario's needs but also maximizes operational agility and efficiency.

1) Global Performance

Global performance was a key requirement in the business scenario, considering the international user base of the Photo Album platform. To address this, we've planned to implement a CDN cache, likely using Amazon CloudFront, to cache content in edge locations around the world. This ensures low-latency access to media content for users, regardless of their geographical location. The use of Amazon CloudFront as a content delivery mechanism greatly enhances global performance. By caching content in edge locations worldwide, it ensures low-latency access

for users across the globe. This comprehensively meets the requirement for global performance, offering users a fast and uninterrupted experience regardless of their location.

4) **Media Processing and Transformation**

The business scenario called for efficient media processing and transformation. Our architecture uses AWS Step Functions to manage media transformation workflows. AWS Step Functions effectively manage media transformation workflows, ensuring fulfillment of the media processing and transformation requirements. These workflows are triggered when media is uploaded, ensuring that images and videos are processed efficiently and in compliance with business scenario demands. When media is uploaded, it undergoes processing via workflows, as specified for images and videos. This design choice ensures that media processing is handled efficiently, aligning with the business's need to optimize content management.

5) **Platform Suitability**

Our architecture effectively fulfills the requirement for platform suitability, as outlined in the business scenario. It is designed to distribute the workload based on the intensity and nature of the task. This distribution is essential for optimizing resource utilization and ensuring that tasks are handled efficiently.

For instance, AWS Lambda is utilized to handle short-lived tasks, ensuring that they are processed swiftly. In contrast, more intense processing tasks can be managed by dedicated resources. This adaptability ensures that the platform can efficiently cater to a wide range of tasks, from lightweight functions to resource-intensive processes.

6) **Decoupling and Load Management**

To meet the requirement for handling varying loads, we've integrated Amazon SQS to decouple application components and manage tasks in a queue. This ensures that the application can efficiently handle varying loads, as seen with SQS integration in the media processing workflows. Decoupling and load management are essential components of the designed architecture, in line with the business scenario's needs.

7) **Futureproofing and Extensions**

Our architecture effectively fulfills the business scenario's requirement for futureproofing and extensions. The design considers potential enhancements and integrations to ensure the platform remains adaptable to changing needs and technological advancements.

AWS recognition services can be seamlessly integrated in the future to enable automatic image and video analysis. This extends the platform's capabilities to offer features like content recognition, tagging, or analysis for improved user experiences and content management.

Additionally, AWS Elemental MediaConvert, though not explicitly shown in the diagram, can be integrated later to provide video transcoding capabilities. This prepares the platform for future needs related to video content, such as format conversions and optimizations for different devices.

VII. Justification for the Designed Architecture (Kelvin)

The chosen serverless/event-driven architecture aligns with the business scenario requirements and offers several key advantages.

1) **Managed Cloud Services and Media Storage:**

The selection of AWS's managed services for infrastructure management aligns with the business scenario's emphasis on reducing in-house system administration. By relying on AWS's expertise, the platform can benefit from efficient and reliable services while minimizing administrative overhead.

The use of AWS S3 buckets for media storage is a robust choice, ensuring that media content is stored securely and efficiently. AWS S3 offers high durability, availability, and scalability, which are essential for handling the growing demands of the Photo Album platform.

2) **Scalability**

By employing AWS Lambda and Amazon API Gateway, our architecture can seamlessly handle increased traffic and load variations, ensuring a responsive and uninterrupted user experience. This scalability is essential to meet the growing user demands outlined in the business scenario.

3) **Serverless/Event-Driven Architecture**

The selection of a serverless and event-driven architecture aligns with the business scenario's emphasis on agility and scalability. By using AWS Lambda, the platform can efficiently respond to various events without the need to manage traditional server infrastructure. This approach allows the platform to scale based on demand, which is critical for addressing user needs and platform growth.

Amazon DynamoDB [7], as the choice for storing metadata, ensures that the platform has a flexible and scalable data storage solution. This is essential for handling the growing amounts of data generated by user interactions and media uploads.

Amazon SQS's role in decoupling application components and managing tasks in a queue ensures that the platform can handle varying workloads without bottlenecks. This is particularly important for media processing and ensuring a smooth user experience.

4) **Global Reach**

The integration of Amazon CloudFront [8] extends our platform's global reach, which is crucial for catering to a worldwide audience. Users from various regions can access content with minimal latency, enhancing the user experience.

5) **Media Processing**

AWS Step Functions [9] facilitate the seamless processing and transformation of media files, ensuring that images and videos are handled efficiently. This is pivotal for a platform that centers around visual media content.

6) **Platform Suitability**

The selection of AWS Lambda for handling short-lived tasks and dedicated resources for intense processing is a sound justification for the architecture. It aligns with the business scenario's requirement for platform suitability, which involves distributing the workload based on task intensity.

AWS Lambda, as a serverless compute service, is well-suited for executing lightweight, event-driven functions. It provides an optimal environment for tasks that require fast, on-demand processing, ensuring that they do not impact the performance of other tasks.

On the other hand, the allocation of dedicated resources for more resource-intensive tasks is a strategic design choice. This ensures that such tasks can be executed efficiently without overloading the serverless components of the architecture.

7) **Decoupling and Load Management**

The adoption of Amazon SQS as part of the serverless/event-driven architecture serves as a robust justification. This design choice ensures the decoupling of application components, which is fundamental to enhancing

the reliability and resilience of the platform. It allows different parts of the application to function independently, reducing potential bottlenecks and ensuring that no component overloads the system.

Load management is equally important, and Amazon SQS assists in managing this effectively. Tasks can be queued and executed at a pace that suits the platform's overall processing capabilities. This architecture not only enhances the platform's ability to manage varying loads but also contributes to its overall robustness, even under peak usage.

8) **Futureproofing and Extensions**

The inclusion of provisions for futureproofing and extensions aligns with the business scenario's focus on staying adaptable and competitive in the ever-evolving digital landscape. By considering AWS recognition services for image and video analysis, the platform can leverage advanced technologies for content recognition and improved user experiences.

The potential integration of AWS Elemental MediaConvert addresses the need for video transcoding, which may become essential as the platform grows and diversifies its multimedia content. It's a proactive step to ensure the platform can efficiently adapt to changing video requirements.

VIII. CONCLUSION

In summation, employing a serverless/event-driven framework on AWS offers immense advantages to businesses in need of scalable and budget-friendly solutions. Tools like AWS Lambda, Amazon SNS, and Amazon SQS seamlessly cater to real-time events, eliminating hands-on management. This lets business proprietors' channel more resources into delivering genuine value to their clientele.

IX. REFERENCE

[1] "Web Application Hosting in the AWS Cloud," Amazon Web Services, Aug. 2021. [Online]. Available: https://d1.awsstatic.com/whitepapers/aws-web-hosting-best-practices.pdf. [Accessed 19 October 2023].

[2] "AWS Serverless Multi-Tier Architectures with Amazon API Gateway and AWS Lambda," Amazon Web Service, Oct. 2021. [Online]. Available: https://d0.awsstatic.com/whitepapers/AWS_Serverless_Multi-Tier_Architectures.pdf. [Accessed 17 October 2023].

[3] "Managing App Performance - AWS Amplify Hosting," docs.aws.amazon.com. [Online]. Available: https://docs.aws.amazon.com/amplify/latest/userguide/ttl.html. [Accessed October 18, 2023]

[4] "Security in Amplify - AWS Amplify Hosting," docs.aws.amazon.com. [Online]. Available: https://docs.aws.amazon.com/amplify/latest/userguide/security.html. [Accessed October 22, 2023]

[5] "Resilience and disaster recovery in Amazon DynamoDB - Amazon DynamoDB," docs.aws.amazon.com. [Online]. Available: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/disaster-recovery-resiliency.html. [Accessed October 02, 2023]

[6] "Cost Optimization Pillar AWS Well-Architected Framework," Amazon Web Service, Oct. 2023. [Online]. Available: https://docs.aws.amazon.com/pdfs/wellarchitected/latest/cost-optimization-pillar/wellarchitected-cost-optimization-pillar.pdf#welcome. [Accessed October 20, 2023].

[7] "Amazon DynamoDB Developer Guide," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html. [Accessed: October 18, 2023].

[8] "Amazon CloudFront Developer Guide," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Introduction.html. [Accessed: October 15, 2023].

[9] "AWS Step Functions Developer Guide," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/step-functions/latest/dg/welcome.html. [Accessed: October 17, 2023].