Serverless/Event-driven Architectural Design Report

Sy Tuan Phan
Lab Section: Saturday 10:00 AM
Swinburne University of Technology
104072029@student.swin.edu.au

Trac Duc Anh Luong
Lab Section: Saturday 10:00 AM
Swinburne University of Technology
103488117@student.swin.edu.au

Tuan Duc Nguyen
Lab Section: Saturday 10:00 AM
Swinburne University of Technology
104189440@student.swin.edu.au

***Abstract*— This report introduces a serverless cloud architecture tailored for the Photo Album application, emphasising the transition to managed cloud services, scalability to meet rising demand, serverless computing, and the replacement of a traditional relational database. The proposed design integrates various AWS services, including Amplify, Cognito, S3, DynamoDB, Lambda, Route 53, SNS, SQS, Rekognition, and API Gateway. The primary goals include optimising the efficiency of photo and video uploads and enhancing global response times. The paper details the architecture's structure, critical use cases, and an overview of the AWS services integrated into the design. Additionally, it offers insights into the rationale behind the architecture, illustrating how it aligns with business requirements and design criteria to deliver a robust and responsive solution for the Photo Album application.**

***Index terms*—Cloud computing, architecture design, serverless, event-driven**

I. Introduction

In this age of technological advancement, digital technologies play an instrumental role in driving business operations to success. Among these emerging technologies, cloud computing is an effective way for business shareholders and software developers to develop and deploy the business foothold to the internet. The notion of cloud computing has revolutionised the online infrastructures of businesses by providing a resource-efficient and economical approach. Due to the blurred line between creating and deploying a web application on-premise and in the cloud, our Photo Album application will leverage the advantages of the cloud using AWS (Amazon Web Services). With AWS's help, managing the cloud application will be a carefree and seamless web operating experience.

A significant upside of AWS is that it offers all the technologies for compiling code, collecting data, and application integration without managing servers [1]. Serverless architectures are driven by calculated events, ensuring the responsiveness and scalability of the website. With serverless, we can shift our focus towards building the architecture without the complexity of servers. The Photo Album application emphasises uploading and sharing captivating media like images and videos. To achieve these goals, the website's main functionalities are included in the proposal:

- Uploading media: Any image (JPG, PNG, WEBP, etc.) or video (MP4, MKV, MOV, etc.) formats can be handled by the system. Upon upload, the media files are resized to create a thumbnail in the dashboard and reformat to standard file extensions like JPG for images and MP4 for video. Image tags are also identified with AI assistance. After processing, the media are ready to be viewed and playback with fast delivery.
- Viewing media: The dashboard will be the main page where uploaded media are displayed. Authenticated users can search and filter desired images or videos based on meta-data associated with the media, like name, description, creating date, keywords, or AI-identified tags.

II. Architecture Overview

A. Architectural Diagram

The diagram below (Figure 1) encapsulates the overview architecture of our proposed solution. In this diagram, we can find all of the AWS services involved and their interactions. Our solution is divided into four tiers: presentation, authentication, logic, and data. We further expand the logic tier into three sections: REST API, image processing, and video processing.

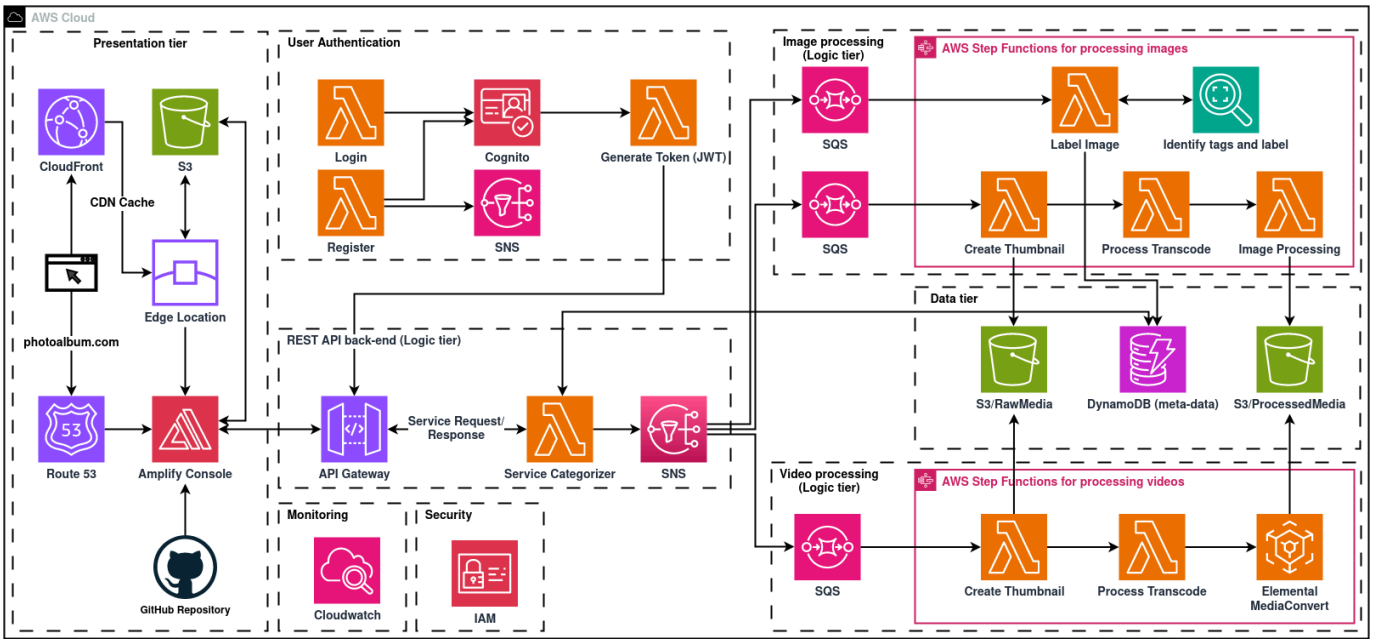

Figure 1: Architectural Diagram

B. *Collaboration Diagrams*

1) *Photos and videos display:*

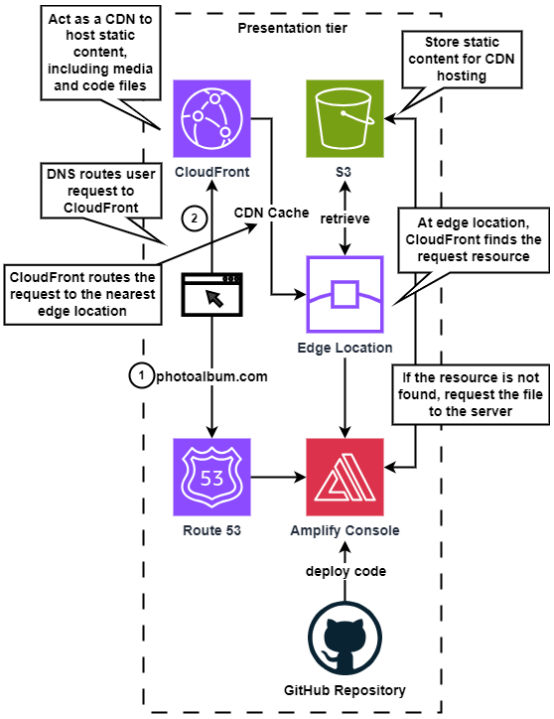

Figure 2: Photos and videos display

In Figure 2, we demonstrate the presentation tier as our AWS infrastructure's content delivery system (CDN). This can be achieved using DNS resolution, AWS Route 53, Content Delivery Services, and AWS CloudFront. There are two paths where the user may access the website's services. Firstly, the user can enter the domain name server into the browser's search box, which will be routed to the website hosted on AWS Amplify through Route 53. Secondly, the website implements a cloud-based media streaming service for content files. The implementation of AWS CloudFront provides the media to users through the nearest edge location. In the scenario where the CDN cache is not available or does not have sufficient data, it

will request the web server to retrieve the necessary media. Our approach ensures fast retrieval speed and efficient content delivery to the users to enhance their experience.

2) *User authentication:*

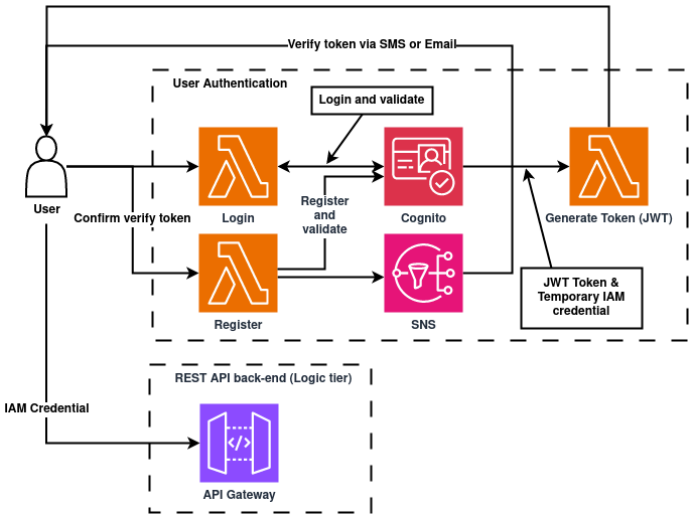

Figure 3: User authentication

Figure 3 contains the logic flow for user authentication, including login for registered users and Registration for new platform users. For Register, the Lambda function triggers SNS and sends the verification token via SMS or Email. The user can then confirm the verification token to create a new user in the User Pool and log in to the website. For login, the user's account and password are validated by Cognito to create a JavaScript Web Token (JWT) and temporary IAM credential for interaction with the web.

3) *Upload and process photos:*

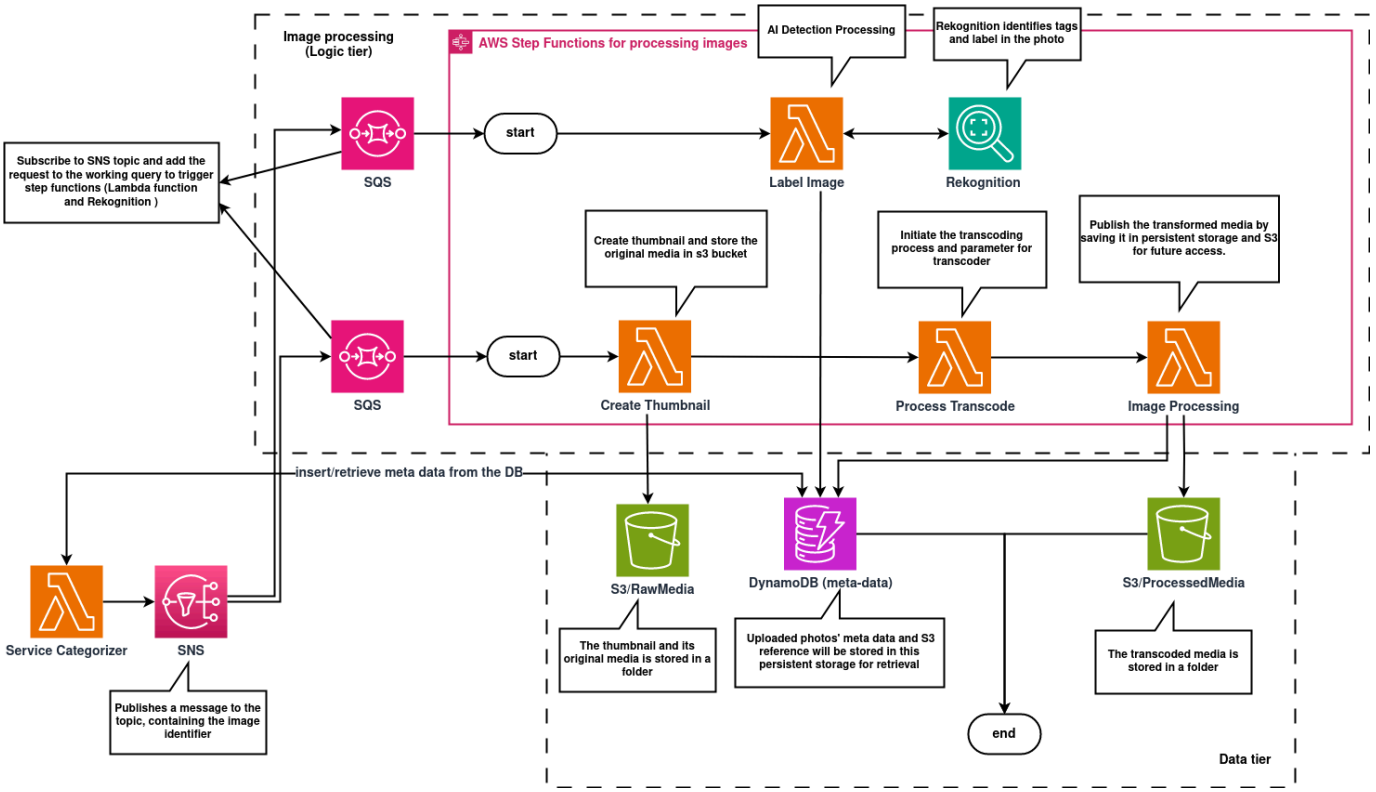

Figure 4: Upload and process photos

The process for uploading and processing photos in AWS begins with users uploading images to an S3 bucket through the photoalbum.com website. Upon upload, S3 triggers an SNS notification, which, in turn, notifies Lambda Functions, initiating the processing workflow. As part of this workflow, the system employs Rekognition to identify objects and tags in the photo, providing valuable metadata while creating a thumbnail. This thumbnail is then stored back in the S3/Raw-Media bucket. Concurrently, the Lambda function initiates the Transcoder to transcode the photo into various formats, such as .png and .jpg, ensuring compatibility across different devices. The resulting transcoded media is stored in a dedicated S3/ProcessedMedia bucket. Subsequently, the Lambda function inserts metadata associated with the photo, including filename, size, and upload timestamp, into DynamoDB for structured information. This orchestrated sequence ensures an efficient and streamlined approach to uploading and processing photos in the AWS environment.

4) *Upload and process videos:*

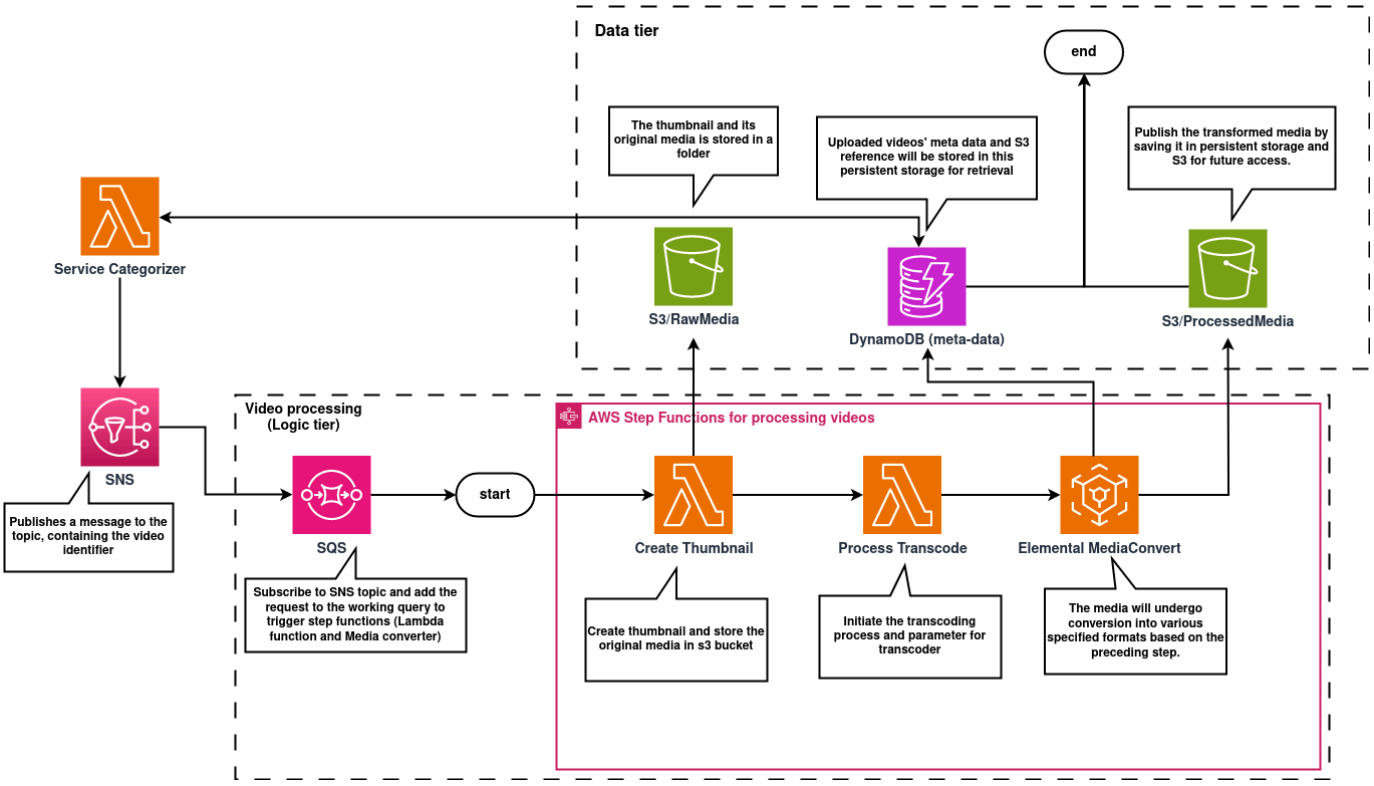

Figure 5: Upload and process videos

The process for uploading and processing videos in AWS follows a systematic flow. Initially, users upload videos to the photoalbum.com website, depositing content into an S3/RawMedia bucket. Subsequently, the AWS server triggers the publication of a message to an SNS topic, conveying the video identifier. An SQS queue subscribes to this SNS topic, adding the request to a working queue to initiate subsequent processing steps. Lambda Functions come into play within a serverless workflow, creating a thumbnail and storing the original media in an S3/RawMedia bucket. Following this, Lambda Functions initiate the transcoding process by providing parameters to the Elemental MediaConvert. Elemental MediaConvert then performs the transcoding, converting the media into multiple specified formats, with the resulting content stored in an S3/ProcessedMedia bucket. Simultaneously, the thumbnail, metadata, and additional metadata for uploaded videos are stored in a DynamoDB table[2]. Finally, the transformed media is published, ensuring its availability in persistent storage, such as S3/ProcessedMedia, for future access and distribution.

C. AWS Services Involved

This section of our proposal demonstrates the details of AWS services implemented in the architecture. In total, we use 14 different services.

1) *CloudFront:*

***Functionality**:* Amazon CloudFront is a content delivery network (CDN) that caches static content, such as images, videos, HTML, and CSS files so that it can be delivered to users from the nearest edge location. This will improve a website's

performance by reducing the time it takes for users to load content. Encryption is another factor when content is delivered through HTTPS and other features, such as origin access identity (OAI) and IP restrictions, to control who can access the content.

***Explanation of choice**:* CloudFront is a crucial component in our architecture for optimizing media content delivery. Caching static assets at edge locations significantly reduces the load times for users globally. This is particularly beneficial for a media-heavy application like the Photo Album, where quick and reliable content delivery is essential for an enhanced user experience.

2) *Route 53:*

***Functionality**:* Amazon Route 53 is a scalable and highly available Domain Name System (DNS) web service. It allows developers to register domain names, route traffic to resources like web servers or load balancers, and distribute domain name queries across global locations. Route 53 also provides health checks to monitor the health and performance of our applications.

***Explanation of choice**:* Route 53 is crucial in our architecture for ensuring high availability and low-latency DNS resolution. It plays a vital role in routing user requests to the nearest edge location, enhancing the overall responsiveness of our Photo Album application.

3) *Amplify:*

***Functionality**:* AWS Amplify is a fully managed service that provides tools and services for building scalable and secure cloud-powered serverless applications. It simplifies the development process by offering features like continuous deployment, authentication, and hosting for web applications.

***Explanation of choice**:* Amplify streamlines the deployment and management of our serverless application by deploying our code base from a GitHub repository to the Amplify Console. Its integration facilitates continuous updates and enhancements to our Photo Album application, ensuring agility and efficiency in the development lifecycle.

4) *S3:*

***Functionality**:* Amazon S3 is an object storage service that stores data in the cloud. It ensures our compliance with durability and cost-effectiveness. S3 can store various data formats, including images, videos, documents, and code (besides images and videos, we also have a bucket in the presentation tier storing HTML, CSS, and JavaScript). It stores data redundantly across multiple data centres for durability to protect it from data loss. S3 also provides various data durability options, such as replication and versioning, to protect data from accidental deletion or corruption.

***Explanation of choice**:* Our application has the perfect use case for S3, as it satisfies the need for ample storage with the flexibility to store objects in any format. The integration among S3, Lambda, and Elemental MediaConvert will be seamless with the automated reprocessed version of uploaded media.

5) *API Gateway:*

***Functionality**:* Amazon API Gateway manages and exposes REST APIs. The API Gateway makes it easy for developers to secure scalable APIs in creating, publishing, maintaining, and monitoring. The unified interface for managing and exposing REST APIs of the API Gateway makes it a popular choice when integrated with other AWS services, such as Amazon Lambda and Amazon S3.

***Explanation of choice**:* For our application, the backend logic mainly utilizes Lambda functions, and the API Gateway will provide various endpoints that users can interact with. This includes endpoints for uploading media, retrieving user data, and other essential functionalities. With Amazon Cognito integration, API Gateway ensures secure access to the APIs. It handles authentication through Cognito, providing JSON Web Tokens (JWTs) to authenticated users. API Gateway automatically scales to handle incoming traffic, ensuring that the Photo Album application remains responsive even during periods of high demand. It leverages the underlying AWS infrastructure to provide low-latency responses to client requests.

6) *Lamda:*

***Functionality**:* Amazon Lambda is a serverless computing service that lets you run code without provisioning or managing servers. It automatically scales based on incoming traffic and executes code responding to predefined events, allowing seamless integration with other AWS services.

***Explanation of choice**:* Lambda is fundamental to our serverless architecture, handling the execution of backend logic triggered by events such as API Gateway requests and S3 events. Its automatic scaling ensures that our application can handle varying workloads efficiently.

7) *SNS:*

Functionality: Amazon Simple Notification Service (SNS) is a fully managed messaging service that enables the distribution of messages to a distributed set of recipients. It supports various communication protocols and can send notifications, alerts, and updates.

Explanation of choice: SNS plays a critical role in our architecture by enabling communication and notification between different components. It facilitates event-driven communication, allowing other system parts to react to specific events and maintain synchronicity.

8) *SQS:*

Functionality: Amazon Simple Queue Service (SQS) is a fully managed message queuing service that enables the decoupling of the components of a cloud application. It ensures reliable and asynchronous communication between different parts of the system.

Explanation of choice: SQS enhances the resilience and scalability of our architecture by decoupling components and managing the flow of messages between them. This ensures that each piece can operate independently, contributing to the overall stability of the Photo Album application.

9) *Cognito:*

Functionality: Amazon Cognito is an identity and access management (IAM) service that can authenticate users, manage user permissions, and track user activity. Cognitio provides a variety of authentication methods, such as username, password, social login, and multi-factor authentication, to authenticate users and verify their identity. It provides a cutting-edge authorization mechanism to control which resource is accessible to whom. The tool adds a layer of protection for sensitive data and prevents unauthorised access. Cognito also comes with various user management features, including user profiles, groups, and roles, to manage our users and their access to data.

Explanation of choice: In our application, Cognito will add authentication to the app by providing legitimate users with JWTs (JSON Web Tokens). The JWT, which contains each user's identity, will be used to verify every request to the API gateway.

10) *Rekognition:*

Functionality: Amazon Rekognition is a fully managed image and video analysis service powered by machine learning. It provides image and video recognition capabilities, face detection, facial analysis, and content moderation. Rekognition can identify objects, people, text, scenes, and activities within images and videos. Its integration with the application allows for automatic tagging and categorization of uploaded media, enhancing the organization and searchability of content.

Explanation of choice: Rekognition is instrumental in augmenting the functionality of our Photo Album application by adding intelligent image analysis. With Rekognition, we can automatically generate tags for uploaded media, making it easier for users to search and categorize their content. This integration enhances user engagement and the overall usability of the application.

11) *DynamoDB:*

Functionality: Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. It is designed to handle large amounts of data and traffic while offering low-latency access to database resources. DynamoDB stores data in tables with flexible schema designs, allowing for efficient data storage and retrieval. It supports document and key-value data models, making it versatile for various application use cases.

Explanation of choice: DynamoDB serves as the database backend for our Photo Album application, offering a scalable and performant solution for storing metadata and user-related information. Its ability to handle large volumes of data and seamless scalability make it an ideal choice for an application dealing with substantial media content and user interactions. The flexible schema design accommodates the diverse data types associated with media files and user profiles.

12) *Elemental MediaConvert:*

Functionality: AWS Elemental MediaConvert is a file-based video transcoding service that provides high-quality video transcoding for various devices and resolutions. It seamlessly integrates with other AWS services, allowing for efficient media content processing and delivery.

Explanation of choice: Elemental MediaConvert is integral to our architecture for processing and converting media files, ensuring compatibility and optimal viewing experiences. Its seamless integration with other services streamlines the media processing workflow in our Photo Album application.

13) *Cloudwatch:*

Functionality: Amazon CloudWatch is a monitoring and observability service that provides data and actionable insights for monitoring applications, responding to system-wide performance changes, and optimising resource utilisation.

***Explanation of choice**:* CloudWatch is essential for monitoring the health and performance of our application. It provides valuable metrics, logs, and alarms that enable proactive management, ensuring the continuous availability and reliability of the Photo Album application.

14) *IAM:*

***Functionality**:* AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS services and resources. It allows the creation and management of AWS users and groups and provides granular control over their permissions.

***Explanation of choice**:* IAM is the backbone of our security infrastructure, ensuring that only authorised entities can access our AWS resources. It provides fine-grained control over permissions, enhancing the overall security posture of the Photo Album application.

D. Requirement Fulfillment

1) *Minimize the need for in-house systems administration with managed cloud services:*

In addressing the imperative to minimize the requirement for in-house systems administration, the architectural design strategically employs managed cloud services to streamline operational efficiency and alleviate administrative burdens.

***DynamoDB for Effortless Database Management**:* The selection of DynamoDB as the primary database solution is pivotal in achieving this objective. DynamoDB, being a NoSQL database, operates serverless, eliminating the need for manual intervention in database management. Its automatic scaling capabilities adjust to real-time traffic patterns dynamically, ensuring optimal performance during peak loads without constant oversight. This enhances operational efficiency and significantly reduces the workload on in-house system administrators.

***API Gateway for Simplified API Management**:*

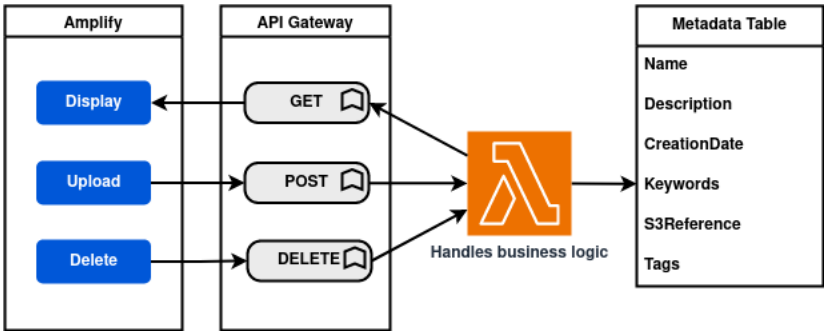

Figure 6: API Gateway design

API Gateway emerges as a central component for API management, providing a fully managed solution encompassing APIs' maintenance, security, and monitoring. By leveraging API Gateway, the company can offload the intricate aspects of API management to a managed service, thereby diminishing the requirement for in-depth systems administration. This allows the company to redirect resources towards core application development activities rather than investing extensively in infrastructure maintenance.

***Amazon S3 for Scalable and Secure Media Storage**:* The utilization of Amazon S3 for storing media aligns seamlessly to minimize in-house systems administration. S3, known for its highly scalable and secure object storage capabilities, eliminates the need for dedicated system administrators to oversee storage infrastructure. Its automatic scaling features accommodate growing storage needs without manual intervention, ensuring the company can focus on application development and user experience rather than intricate storage management.

2) *Cope with demand doubling every 6 months:*

In addressing the imperative to effectively handle the doubling of demand every six months, the architectural design strategically integrates Lambda functions as a critical component to accommodate varying workloads.

***Lambda Functions for Dynamic Workload Handling**:* Lambda functions emerge as crucial elements in dynamically managing the increasing demand. With the expectation of the user base doubling every six months, Lambda's intrinsic ability to scale seamlessly becomes instrumental. The functions automatically adjust to match concurrent requests, ensuring optimal resource utilization. This serverless approach eliminates manual intervention in scaling the infrastructure, allowing the system to efficiently handle a surge in incoming requests without compromising performance.

***DynamoDB's Auto-Scaling for Database Optimization**:* DynamoDB's auto-scaling feature adds another layer of adaptability to the system. This feature utilizes the AWS Application Auto Scaling service to adjust provisioned throughput capacity dynamically based on real-time traffic patterns. By doing so, DynamoDB prevents throttling during sudden spikes in demand, optimizing database performance. This auto-scaling capability ensures that the database can efficiently scale up its read and write capacity to accommodate increased traffic without incurring performance degradation. DynamoDB, operating in tandem with Lambda functions, forms a robust foundation for handling the anticipated growth in demand.

***API Gateway's Dynamic Scaling for API Traffic**:* API Gateway seamlessly complements Lambda's capabilities by dynamically scaling its capacity to accommodate the growing API traffic. As the user base expands, API Gateway adjusts automatically to handle varying traffic levels, ensuring a smooth and scalable interaction between the frontend and backend services. This dynamic scaling feature adds resilience to the system, allowing it to gracefully handle surges in traffic without compromising overall reliability. API Gateway's ability to efficiently manage API workloads aligns with the overarching goal of coping with the escalating demand.

3) *Adopt a serverless/event-driven solution:*

Adopting a serverless and event-driven solution in the architectural design is a strategic choice, aiming to enhance efficiency, responsiveness, and resource utilization. The core elements contributing to this approach include hosting the static website in an S3 bucket and implementing backend logic through Lambda functions [3].

***Lambda's Serverless Architecture and Automatic Scaling**:* Lambda's serverless architecture plays a pivotal role in achieving the event-driven and serverless nature of the solution. This architecture allows for automatic scaling, ensuring that computing resources are dynamically allocated based on demand. The serverless paradigm eliminates the need for provisioning and managing servers manually, promoting a more agile and cost-effective system. Lambda functions scale seamlessly as the workload fluctuates, adapting to varying demand levels and optimizing resource utilization.

***Event-Driven Nature of Lambda for Immediate Responses**:* Lambda's event-driven nature is a crucial feature aligning with the company's preference for a highly responsive system. The functions respond immediately to specific triggers, allowing for real-time processing and execution of tasks. This event-driven architecture enhances the system's agility and responsiveness, ensuring that it can swiftly adapt to changing conditions and efficiently handle diverse tasks triggered by specific events [4].

***Cost-Effective and Scalable Solution**:* The serverless and event-driven approach, leveraging Lambda and S3, contributes to a cost-effective and scalable solution. The elimination of traditional server maintenance reduces operational costs, while automatic scaling ensures optimal resource usage without incurring unnecessary expenses. This combination creates a solution that can efficiently scale with the application's growth and adapt to variable workloads, aligning with the company's goal of cost-effectiveness and scalability.

4) *Replace the relational database with faster and cheaper options:*

The decision to replace the traditional relational database with DynamoDB aligns with the company's objective of achieving a faster and more cost-effective data storage solution. DynamoDB's Key Features for Enhanced Performance: DynamoDB's key-value data model, automatic scaling, and low-latency performance make it a suitable alternative to traditional relational databases. The key-value model allows for efficient storage and retrieval of simple metadata related to photos and videos. Automatic scaling ensures that the database dynamically adjusts its capacity based on real-time traffic patterns, preventing performance bottlenecks during periods of increased demand. This flexibility contributes to a more responsive and efficient data storage solution.

***Cost-Effectiveness of DynamoDB**:* DynamoDB's pay-as-you-go pricing model ensures cost-effectiveness, particularly for applications with varying data storage requirements. Unlike fixed-schema relational databases, DynamoDB allows the application to scale its throughput capacity based on demand, optimizing costs during periods of lower activity. This aligns with the company's goal of achieving a cost-effective solution without compromising performance.

5) *Improve global response times for users outside Australia:*

Addressing the challenge of slow global response times is a crucial aspect of architectural design, and Amazon CloudFront plays a central role in achieving this goal.

***CloudFront's Global Content Delivery**:* CloudFront's role is to cache content at edge locations worldwide, including static web files and user-uploaded media. By doing so, CloudFront significantly reduces latency for users located outside Australia. Integrating S3 ensures that content is delivered securely and efficiently to users, creating a more responsive and globally accessible application. This strategic use of content delivery networks aligns with the company's global goal of enhancing the user experience.

Efficient Content Delivery with CloudFront*:* CloudFront's ability to distribute content from edge locations closest to users ensures faster delivery times, improved load times for web pages, and an overall enhanced user experience. The dynamic integration with S3 allows for on-the-fly processing and customization of content, further optimizing the delivery process. This approach ensures that users, regardless of their geographical location, experience minimal latency when accessing the application, contributing to improved global response times.

6) *Handle video media:*

Incorporating Elemental MediaConvert into the architecture addresses the need to handle video media efficiently.

Elemental MediaConvert's Video Transcoding Capabilities*:* Elemental MediaConvert brings video transcoding capabilities to the architecture, allowing the system to process and adapt video content efficiently to meet the application's requirements. The service supports various video formats, ensuring compatibility and flexibility in handling diverse media content. Its scalability ensures that the application can seamlessly handle varying workloads related to video processing. [5] Fully Managed and Cost-Effective Solution: The fully managed nature of Elemental MediaConvert and its pay-as-you-go pricing model contribute to a cost-effective solution, particularly during lower video processing demand periods. The integration of Elemental MediaConvert into the existing architecture is seamless, ensuring that video media can be incorporated into the application without compromising performance and resource utilization. This addition ensures that the application remains versatile and can cater to diverse media types, aligning with the evolving needs of the Photo Album application.

Efficient Parallel Processing with S3 Event Notifications*:* S3 Event Notifications play a crucial role in triggering the processing pipelines. When media is uploaded to the S3 bucket, S3 Event Notifications initiate the corresponding SNS notifications, which, in turn, fan out to multiple SQS queues. This mechanism allows for efficient parallel processing of different media handling aspects, ensuring that tasks can be added or modified independently without affecting the overall system. Ensuring Efficiency and Extensibility: The extensible and decoupled architecture ensures efficiency in handling uploaded media. New processing tasks can be seamlessly integrated into the system by adding new SQS queues and connecting them to the SNS topics. This modular design facilitates scalability, allowing the system to adapt to evolving requirements without disrupting existing functionalities. The combination of SNS, SQS, and S3 Event Notifications creates a robust architecture for automatically processing uploaded media in a scalable, extensible, and efficient manner.

Authenticate users to control access to the application*:* Amazon Cognito is the solution to ensure secure user authentication and authorisation.

Amazon Cognito for Secure User Authentication*:* Amazon Cognito provides a fully managed solution for user authentication and authorization. Its integration with API Gateway ensures that only authenticated users can access backend services, aligning with the business's security requirements. Cognito supports industry-standard protocols like OAuth 2.0 and OpenID Connect, providing a secure and scalable authentication solution for the application.

Secure Access Control with API Gateway Integration*:* The integration with API Gateway ensures that only authenticated users with valid tokens can access the application's backend services. This secure access control mechanism prevents unauthorized access and safeguards sensitive user data. Cognito's user pools enable the management of user identities and provide features like multi-factor authentication, enhancing the overall security posture of the application.

Fully Managed and Scalable Authentication Solution*:* Amazon Cognito's fully managed nature ensures that the complexity of user authentication is abstracted away, allowing the development team to focus on building application features. Additionally, Cognito scales seamlessly with the user base, ensuring the authentication system remains robust and responsive even as the application experiences growth. The selection of Amazon Cognito aligns with the business's emphasis on security, simplicity, and scalability in user authentication.

7) *Automatically process uploaded media with an extensible and decoupled architecture:*

The architecture employs a sophisticated and extensible approach to meet uploaded media's dynamic processing demands, combining Amazon S3, Simple Notification Service (SNS), Simple Queue Service (SQS), and AWS Lambda functions.

SNS (Simple Notification Service) and SQS (Simple Queue Service) are strategically employed to process uploaded media automatically with an extensible and decoupled architecture. SNS and SQS for Extensibility and Decoupling. The architecture adopts a "fan-out" approach using SNS and SQS, creating an extensible and decoupled system for processing uploaded media. SNS serves as a messaging service, broadcasting notifications to multiple SQS queues. SQS, in turn, facilitates parallel processing of different tasks related to media uploads. This decoupled design ensures that each processing task operates independently, promoting modularity and extensibility.

Upon a customer uploading photos or videos into the designated S3 bucket, S3 Event Notifications are triggered. This event is then broadcast to an SNS topic, serving as the central hub for initiating downstream processes. In this context, SNS enables a one-to-many broadcast, ensuring that multiple services can subscribe to and act upon these notifications concurrently.

The photo processing pipeline, designed for agility and parallelization, features multiple SQS queues subscribing to the SNS topic. Each SQS queue is associated with a specific type of photo processing task, such as thumbnail generation, AI-based image analysis, or other custom operations. Lambda functions are then connected to these SQS queues, polling for new messages and executing the associated processing tasks independently. This "fan-out" architecture allows for parallelized processing of photos, ensuring that various operations can be performed concurrently without creating bottlenecks.

For instance, one Lambda function might leverage Amazon Rekognition to identify tags in the uploaded photos, while another generates mobile-friendly thumbnails. This modular design ensures flexibility and extensibility; as new processing tasks are envisioned, additional SQS queues can be introduced and subscribed to the SNS topic effortlessly. The video processing pipeline follows a similar architecture, albeit with a single SQS queue for the present transcoding requirements. However, anticipating future advancements and complex video processing tasks, the architecture is designed to seamlessly accommodate the addition of more SQS queues for specialized processing needs.

This decoupled design enhances the system's resilience and allows easy modification and expansion. Should a processing task fail or encounter issues, it won't impact the entire system, as the decoupled nature ensures that each task operates independently. Moreover, this architecture facilitates load balancing, as multiple Lambda functions can simultaneously process tasks, preventing the system from becoming overwhelmed during periods of high demand. The reliability and scalability of this architecture make it well-suited to handle the intricacies of media processing, providing an efficient and future-proof solution for the Photo Album application.

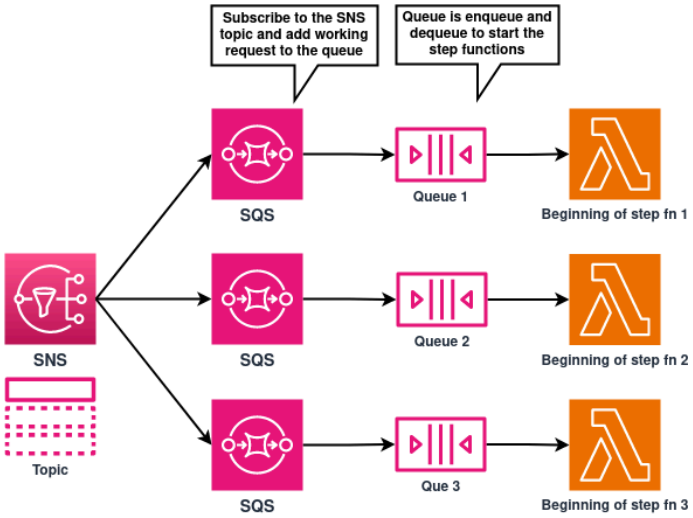

Figure 7: Fan-out approach for decoupling

8) *Authenticate users to control access to the application:*

User authentication is critical to the Photo Album application, ensuring only authorized users can access and interact with the system. The chosen solution for user authentication is Amazon Cognito, a fully managed service designed to simplify the implementation of robust authentication and user management functionalities.

Amazon Cognito Overview*:* Amazon Cognito offers comprehensive features, including user pools, identity pools, and authentication flows, making it a suitable choice for securing the Photo Album application.

User Pools*:* User pools in Amazon Cognito act as users' directories, providing user registration, authentication, and account recovery services. In the Photo Album application context, a user pool is created to manage the user base, allowing users to sign up, sign in, and securely interact with the application. Hosted UI for User Authentication: Amazon Cognito includes a Hosted UI that serves as a pre-built authentication server. This UI lets users sign up and sign in to the application securely. The Hosted UI is customizable, allowing for a seamless integration of the authentication process into the application's user interface.

***Token-based Authentication**:* Upon successful authentication, Amazon Cognito issues JSON Web Tokens (JWTs) to authenticated users. These tokens contain user identity information and are automatically included in the authenticated user's subsequent API requests. This token-based approach ensures that only valid and authenticated users can access the backend services through API Gateway.

***Security Features**:* Amazon Cognito incorporates robust security features, including multi-factor authentication (MFA) and compromised credential protection. MFA adds an extra layer of security by requiring users to provide multiple forms of identification. Compromised credential protection helps identify and respond to compromised user credentials, enhancing the overall security posture of the application.

***Scalability**:* Cognito is designed to scale effortlessly with the application's user base. It provides a free tier for the first 50,000 active monthly users, and beyond that, the pricing is based on the number of active users. This scalability ensures the authentication system can handle the current user load and seamlessly scale as the application grows.

***Cognito User Pool Authorizer with API Gateway**:* API Gateway is the frontend for the backend services, and access to these services is controlled using Cognito User Pool Authorizer.

***JSON Web Token Verification**:* The Cognito User Pool Authorizer is configured in the API Gateway to validate incoming requests' JWTs. It ensures that each request has a valid token issued by Amazon Cognito, verifying the authenticity of the user.

***Authorization Checks**:* With Cognito User Pool Authorizer, API Gateway can perform fine-grained authorization checks based on user roles and permissions. This allows the application to control access to specific resources and functionalities based on the authenticated user's role.

***Integration with Backend Services**:* Once the authentication and authorization checks are successful, API Gateway forwards the request to the appropriate Lambda functions or backend services, allowing authenticated users to interact with the core features of the Photo Album application.

***Comparison with Alternative Solutions**:* While a custom authentication solution using a Lambda authorizer could offer more flexibility, it comes with trade-offs in development effort, security, and ongoing maintenance. A custom solution would require the development of custom authentication and authorization codes, potentially introducing security risks if not implemented correctly. Each API request would also incur an extra Lambda call, increasing operational costs.

9) *Conclusion:*

The proposed architecture leverages a combination of managed cloud services to address the identified problems and requirements. It aligns with the business goals of minimizing in-house systems administration, accommodating future growth, adopting a serverless/event-driven approach, and improving efficiency and cost-effectiveness. The selected services, including DynamoDB, Lambda, API Gateway, S3, CloudFront, Elemental MediaConvert, SNS, SQS, and Cognito, work cohesively to create a scalable, reliable, and secure infrastructure for the Photo Album application, meeting the company's evolving needs.

III. Design Rationale

A. Design criteria

This segment outlines how the suggested services align with the application's performance and authorisation standards.

1) *Performance and Scalability:*

CloudFront: Caches content globally, enhancing response times for users worldwide.

Lambda: Offers automatic scaling and event-driven execution, ensuring seamless handling of peak demands, such as media uploads and processing.

API Gateway: Efficiently manages incoming API requests [6], automatically scales to handle varying traffic levels, and distributes requests across multiple backend services.

Simple Storage Service (S3): Provides efficient and reliable storage, handling massive data and concurrent requests globally through CloudFront, ensuring optimised performance and reduced latency.

DynamoDB: Efficiently stores metadata with a simple key-value data model, providing cost efficiency and faster speeds than traditional relational databases. Horizontal scaling allows seamless scaling as storage demand grows.

Cognito: Offers a fully managed authentication service that scales to support millions of users, aligning with the preferred managed services.

Rekognition: Offers a fully managed and scalable solution for identifying tags from uploaded photos using machine learning, simplifying the tagging process.

Simple Queue Service (SQS): Decouples processing pipelines, preventing overload and job loss, enhancing system reliability.

Simple Notification Service (SNS): Decouples processing pipelines, enabling parallel processing to improve performance, creating a fan-out architecture for extensibility.

2) *Reliability:*

Route 53: Provides a globally distributed network of DNS servers, ensuring consistent and reliable routing using a custom domain name.

API Gateway: Handles high traffic volumes without compromising performance, incorporating features like API throttling and authentication for a secure and reliable user experience.

S3: Provides vast storage capacity and supports various media formats, ensuring secure and reliable storage for uploaded photos and videos. DynamoDB: Offers fast and reliable storage for metadata with a simple key-value data model and horizontal scaling.

SQS: Decouples processing pipelines, ensuring a robust, fault-tolerant system by preventing overload and job loss.

SNS: Decouples processing pipelines, enabling parallel processing of multiple versions of source media files to improve performance.

3) *Security:*

Cognito: Enables robust security with token-based authentication, claims-based tokens, and encryption for sensitive data. Supports Multi-Factor Authentication (MFA) for added security.

Lambda: Offers robust security with IAM roles, fine-grained access control, and data encryption, ensuring data privacy.

S3: Incorporates strong security measures such as access control lists (ACLs), bucket policies, and server-side encryption for data protection.

DynamoDB: Provides data encryption at rest and during transit, along with IAM policies and fine-grained access control for limiting access to tables and data.

API Gateway: Uses IAM roles, unique authentication methods, and AWS Cognito for request authentication and authorisation, effectively restricting API access based on user identities and permissions.

Overall, the selected AWS services demonstrate a robust combination of performance, scalability, reliability, and security, aligning with the design criteria for the Photo Album application. The integrated features of these services contribute to an architecture that can efficiently handle dynamic workloads, ensure data integrity, and provide a secure user experience.

4) *Estimated Cost:*

The pricing structure and service configuration pertinent to our system are delineated herewith. Our operational assumptions are grounded in the Asia Pacific (Sydney) region. Presented in the ensuing table is an estimation predicated on an anticipated daily visitation of 500 users, encompassing both domestic and international access [7], [8], [9], [8], [10], [11] .

Service	Configuration summary	Monthly	First 12 months total
Route 53	Number of hosted zones: 1 IP (CIDR) blocks: 1,000	51.29 USD	615.48 USD
CloudFront	Price for countries outside of Australia, such as Canada, Asia Pacific, Europe,… For example: USA Tiered price for: 3000 GB 3000 GB x 0.0850000000 USD = 255.00 USD Total tier cost = 255.00 USD (Data transfer out to internet from United States) Data transfer out to internet cost: 255.00 USD 3000 GB x 0.02 USD = 60.00 USD (Data transfer out to origin from United States) Data transfer out to origin cost: 60.00 USD 10,000 requests x 0.000001 USD = 0.01 USD (HTTPS requests from United States) Requests cost: 0.01 USD 255.00 USD + 60.00 USD = 315.01 USD (Total cost United States) CloudFront price United States (monthly): 315.01 USD	472.50 USD	5670 USD
Lambda	Standard Lambda Architecture: x86 Number of requests: 310,000 per month (30.41 days * 500 daily users * 15.5 view/upload requests) Duration of each request: 500ms Allocated memory: 128 MB Ephemeral storage: 512 MB Lambda@Edge: Number of requests: 6,020,000 per month (30.41 days x 500 monthly users x 10 daily view requests x 20 photos/videos requested per view) Duration of each request: 500ms Allocated memory: 128MB	22.40 USD	268.8 USD
API Gateway	API type: REST API Number of requests: 310,000 per month (30.41 days * 500 daily users * 15.5 view/upload requests) Caching: None	1.08 USD	12.96 USD
S3	S3 tier: Standard Standard storage amount: 4561.5 per month (500 daily users * 5 daily photo uploads * 30.41 days * 2 MB per photo + 500 daily users * 0.5 daily video uploads * 30.41 days * 500MB per video) Data will be moved into S3 through PUT, COPY, POST requests. Average object size = 47.2MB ((2MB * 5 + 500MB * 0.5) / 5.5) PUT, COPY, POST, LIST requests into S3 = 100,355 (500 users * 5.5 requests per user * 30.41 days) GET requests from S3 standard: 3,649,200 (500 users * 10 requests * 20 photos/videos retrieved per request * 30.41 days) Data returned or scanned by S3: 0 Inbound Data Transfer from Internet (free) Outbound data transfer to CloudFront (free)	116.20 USD	1394.4 USD
DynamoDB	DynamoDB on-demand capacity (standard table class) Data storage size(100GB) Number of write 100,355 = 500 users per day * 30.41 days * 5.5 writes per days each user Number of read: 182,460 = 500 users per day * 30.41 days per month * 10 reads per day each user Monthly read cost (Monthly): 0.07 USD = 273,690.00 total read request units x 0.0000002846 USD DynamoDB data storage cost (Monthly): 28.50 USD Monthly write cost (Monthly): 1.43 USD = 1,003,550.00 total write request units x 0.0000014231 USD Total Monthly cost: 30.00 USD	30.00 USD	360 USD
SNS	Number of requests/notifications: 5 photos per user * 0,5 video per user * 500 users monthly * 30.41 days per month = 100,355 requests/notifications 100,355 requests - 1000000 free tier requests = -899,645.00 billable SNS requests per month Max (-899645.000000 requests, 0 requests) = 0.00 requests 100,355 notifications x 0.00 USD = 0.00 USD (SQS Notifications cost) SNS Requests and Notifications cost (monthly): 0.00 USD	0.00 USD	0.00 USD

SQS	Number of photo queue requests: 5 photos per user * 500 users monthly * 30 days monthly * 3 SQS requests per SNS requests = 225,000 requests Number of video queue requests: 0.5 video per user * 500 users monthly * 30 days monthly = 7,500 requests Number of queue requests: 225,000 + 7,500 = 232,500 requests 0.2325 requests per month x 1000000 multiplier for million = 232,500.00 total standard queue requests Tiered price for: 232500.00 requests 232500 requests x 0.0000000000 USD = 0.00 USD Total tier cost = 0.0000 USD (Standard queue requests cost) Total SQS cost (monthly): 0.00 USD	0.00 USD	0.00 USD
Rekognition	Number of images processed with labels API calls per month: 91,230 91,230 = 500 users per day x 5 images per use	0.00 USD	0.00 USD
Elemental MediaConvert	Output usage: 22807 22807 = 500 users per day x 0.5 video per day x 5 minutes per video x 30,41 days	581.58 USD	6978.96 USD
Cognito	Number of Monthly Active Users(MAUs): 500 users monthly * 30.41 days monthly = 15,205 MAUs 15,205 MAUs x 0.75 SAML or OIDC federation requests = 11,403.75 SAML or OIDC federation MAU requests 11,403.75 SAML or OIDC federation MAUs - 50 free SAML or OIDC federation MAU requests per month = 11,353.75 billable SAML or OIDC federation MAU requests Max (11353.75 billable SAML or OIDC federation MAU requests, 0 minimum billable SAML or OIDC federation MAU requests) = 11,353.75 total billable SAML or OIDC federation MAU requests 11,353.75 MAUs x 0.015 USD = 170.31 USD (SAML or OIDC federation MAU requests) SAML or OIDC federation cost (monthly): 170.31 USD 15,205 MAUs - 50000 free MAU requests per month = -34,795.00 billable MAU requests Max (-34795.000000 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 Advanced security feature cost (monthly): 0 USD Cognito MAU cost (monthly): 170.31 USD	170.31 USD	2043.72 USD
Amplify	3000 GB data storage on CDN per month Build and deploy: 1000 build minutes per month Data transfer out to internet: 15 GB per month Request count (SSR): 500,000 requests per month Request duration (SSR): 100 GB-hours per month Backend: API Gateway, Lambda, DynamoDB, S3, Cognito, SNS, SQS, Rekognition, MediaConvert	0.00 USD for the first 12 months, then 1,308.58 USD per month	15,702.96 USD After first 12 months
IAM	Number of users: 500 Number of groups: 10 Number of roles: 20 Number of policies: 50	0.00 USD	0.00 USD
CloudWatch	30 custom metrics Dashboards: 3 custom dashboards Alarms: 10 standard resolution alarms Logs: 5 GB data ingestion, 5 GB data storage, 5 GB data scanned by Logs Insights queries Events: 100,000 events Contributor Insights: 1 rule, 1 million log events Synthetics: 100 canary runs	0.00 USD for the first 12 months, then 9.09 USD per month	109.08 USD After first 12 months
Total		**1,474.28 USD (After first 12 months : 2791.95 USD)**	**17344.32 USD (After first 12 months : 33156.36 USD)**

Figure 8: Cost estimation table

B. Alternative Solutions Comparison

1) *Relational Database vs Non-relational Database:*

In the initial stages of development, the Photo Album website was built upon an architecture centred around EC2 instances and RDS, utilising the capabilities of a relational database service from AWS. As the company progresses towards a migration to a serverless architecture, two noteworthy alternatives offered by AWS stand out: DynamoDB, a NoSQL database service, and AWS Aurora Serverless, a serverless version of the traditional relational database [12].

***Performance and scalability**:* AWS Aurora Serverless performance scales vertically to meet the needs of an application by adding more resources to maintain performance when the application's load increases and removing resources accordingly when the load decreases. On the other hand, DynamoDB scales horizontally throughput capacity to meet the demands of our applications. As the number of requests increases, DynamoDB increases the capacity units allocated to the meta-data table. This enables DynamoDB to maintain consistent performance as the size of our data grows. The company expects demand growth to double every six months for the next 2-3 years. Therefore, scaling horizontally is a better option, making DynamoDB a better choice.

***Cost-effectiveness**:* DynamoDB's pricing model is based on throughput and storage, providing a pay-as-you-go structure. With its serverless nature, users only pay for the resources consumed, offering a cost-effective solution for applications with varying workloads. Conversely, Aurora Serverless necessitates provisioning a minimum capacity, which could result in needless expenses during low traffic.

***Data model and flexibility**:* Our Photo Album application requires us to store media and their meta-data in the database. NoSQL is more suitable given the simple data structure and relationship using key-value pairs. AWS Aurora Serverless adheres to the relational database model, which, while ideal for structured data, may introduce complexities when dealing with diverse or evolving data structures. DynamoDB's NoSQL flexibility, however, offers seamless adaptability to the growing data demands of the Photo Album application.

***Ease of development and maintenance**:* DynamoDB's seamless integration with serverless architectures eases the development process and significantly reduces maintenance overhead. The straightforward schema design and automated scaling make DynamoDB the preferred choice for creating an agile and efficient serverless architecture for the Photo Album website. AWS Aurora provides a familiar SQL environment, but managing intricate database structures will be a significant issue regarding our application's development and maintenance life cycle.

Considering these factors, DynamoDB and NoSQL solutions generally appear to be a well-suited and cost-effective choice for storing the simple metadata of the Photo Album website. It provides flexibility, scalability, and performance without the overhead of managing complex relational structures.

2) *Elastic Transcoder vs Elemental MediaConvert:*

Amazon Elastic Transcoder and Amazon Elemental MediaConvert are cloud-based media transcoding services offered by Amazon Web Services (AWS). They both allow us to convert videos from one format to another, but they have different features and target other use cases.

***Flexibility and performance**:* Amazon Elemental Media Convert provides a wide range of presets, configurations, and advanced features while delivering high-quality transcoding for professional media workflows [13]. On the other hand, Amazon Elastic Transcoder offers a limited range of presets and configurations while handling basic transcoding tasks efficiently [14].

***Encoding speed**:* Amazon Elastic Transcoder is suitable for moderate transcoding workloads, while Elemental MediaConvert offers faster encoding speeds for demanding workloads.

***Cost-effectiveness**:* Amazon Elastic Transcoder is generally lower pricing for basic transcoding needs. On the other hand, Elemental MediaConvert typically has higher pricing due to advanced features and performance.

After thorough consideration, we found that Elastic Transcoder suits developers and end users who require a simple and cost-effective transcoding solution for basic web and mobile applications. Elemental MediaConvert is better suited for professional media companies that demand high-quality transcoding for digital media workflows. Since our website is for photos and videos, we chose Elemental MediaConvert for media encoding and processing.

3) *AWS Fargate vs AWS Lambda:*

Amazon Fargate and AWS Lambda are serverless computing services offered by Amazon Web Services (AWS). They both allow us to run code without provisioning or managing servers. However, they have different features and target other use cases.

***Type of service and execution model**:* AWS Fargate is a container-based serverless computing service [15]. It lets you run Docker containers without having to provision or manage servers. Fargate is a good choice for stateless applications that quickly scale up or down. AWS Lambda is a function-as-a-service (FaaS) serverless computing service [16]. It lets you run code as functions triggered by events, such as file uploads or API requests. Lambda is a good choice for applications that are short-lived and event-driven. Our infrastructure requires the support of triggers to initiate services like SQS subscriptions, DynamoDB, and S3 events. The company is leaning towards an event-driven approach, so Lambda is more suitable as a computing option.

***Cost-effectiveness**:* Fargate charges based on vCPU and memory usage, while Lambda charges based on the number of requests and execution time. As our application is stateless and is triggered by events like API requests and media file uploads, Lambda is a better choice.

***Ease of use**:* Fargate is more complex to use than Lambda. It requires you to have a good understanding of Docker containers and how to manage them. As we want to implement a seamless architecture, the complexity of managing the codebase and containers should be reduced if necessary.

IV. Conclusion

In summary, the proposed architecture for the Photo Album application represents a sophisticated and efficient system designed to meet the specific needs outlined by the company. The architecture addresses critical challenges and aligns with strategic objectives by leveraging a serverless and event-driven approach on AWS.

Adopting AWS Lambda, Amazon S3, and API Gateway is pivotal in achieving a scalable and cost-effective solution. The serverless nature of Lambda ensures automatic scaling, optimizing resource utilization based on real-time demand. This enhances the system's responsiveness and eliminates manual management, allowing the development team to focus on creating customer value rather than infrastructure maintenance.

DynamoDB, as a NoSQL database, provides a fast and cost-effective alternative to traditional relational databases. Its key-value data model and automatic scaling improve performance and efficiency in storing metadata associated with photos and videos. This choice aligns seamlessly with the company's goal of achieving a more responsive and resource-efficient data storage solution.

To cope with the anticipated doubling of demand every six months, the architecture incorporates Lambda's ability to manage concurrent requests and scale dynamically. This ensures that the system can seamlessly handle the expected growth in user base without compromising performance.

Efforts to enhance global response times for users outside Australia involve the strategic integration of Amazon CloudFront. By caching content at edge locations worldwide, CloudFront significantly reduces latency, contributing to a more responsive and globally accessible application. This aligns with the overarching goal of providing an optimal user experience on a global scale.

The inclusion of Elemental MediaConvert addresses the specific requirement of handling video media efficiently. Its broad format support, scalability, and pay-as-you-go pricing model ensure a robust solution for transcoding videos, enhancing the application's versatility without sacrificing cost-effectiveness.

The architecture adopts an extensible and decoupled approach using SNS and SQS to process uploaded media automatically. This design allows for parallel processing of different tasks, ensuring efficiency and extensibility. S3 Event Notifications further streamline the processing pipelines, providing a scalable and modular solution that can adapt to evolving requirements.

Lastly, to ensure secure user authentication and authorization, Amazon Cognito is selected, providing a fully managed and scalable solution. Its integration with API Gateway ensures that only authenticated users can access backend services, aligning with the company's emphasis on security and simplicity in user authentication.

In conclusion, the comprehensive architecture outlined above positions the Photo Album application for success, combining the strengths of various AWS services to create a system that is responsive, scalable, cost-effective, and well-aligned with the business's objectives.

References

[1] AWS, "Serverless Computing – Amazon Web Services". Accessed: Nov. 21, 2023. [Online]. Available: https://aws.amazon.com/serverless/

[2] AWS, "Amazon DynamoDB features". Accessed: Nov. 24, 2023. [Online]. Available: https://aws.amazon.com/dynamodb/features/

[3] A. W. Services, "Serverless logic tier - AWS Serverless Multi-Tier Architectures with Amazon API Gateway and AWS Lambda". [Online]. Available: https://docs.aws.amazon.com/whitepapers/latest/serverless-multi-tier-architectures-api-gateway-lambda/serverless-logic-tier.html

[4] A. W. Services, "Lambda function scaling - AWS Lambda". [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/lambda-concurrency.html

[5] A. oshi, "AWS MediaConvert: The Powerful Alternative to Elastic Transcoder". [Online]. Available: https://www.cloudthat.com/resources/blog/aws-mediaconvert-the-powerful-alternative-to-elastic-transcoder

[6] AWS, "Amazon API Gateway features". Accessed: Nov. 24, 2023. [Online]. Available: https://aws.amazon.com/api-gateway/features/

[7] AWS, “Amazon API Gateway Pricing | API Management | Amazon Web Services”. Accessed: Nov. 25, 2023. [Online]. Available: https://aws.amazon.com/api-gateway/pricing/

[8] AWS, “Pricing | Amazon Cognito | Amazon Web Services (AWS)”. Accessed: Nov. 25, 2023. [Online]. Available: https://aws.amazon.com/cognito/pricing/

[9] AWS, “AWS Lambda – Pricing”. Accessed: Nov. 25, 2023. [Online]. Available: https://aws.amazon.com/lambda/pricing/

[10] N. Dimitri, “Pricing cloud IaaS computing services”, *Journal of Cloud Computing*, 2020, doi: 10.1186/s13677-020-00161-2.

[11] M. Al-Roomi, S. Al-Ebrahim, S. Buqrais, and I. Ahmad, “Cloud Computing Pricing Models: A Survey”, *International Journal of Grid and Distributed Computing*, p. 93, 2013, Accessed: Nov. 25, 2023. Available: https://www.academia.edu/36930119/Cloud_Computing_Pricing_Models_A_Survey

[12] AWS, “Relational vs Nonrelational Databases - Difference Between Types of Databases - AWS”. Accessed: Nov. 22, 2023. [Online]. Available: https://aws.amazon.com/compare/the-difference-between-relational-and-non-relational-databases/

[13] AWS, “AWS Elemental MediaConvert”. Accessed: Nov. 22, 2023. [Online]. Available: https://aws.amazon.com/mediaconvert/

[14] AWS, “AWS | Amazon Elastic Transcoder - Media & Video Transcoding in the Cloud”. Accessed: Nov. 22, 2023. [Online]. Available: https://aws.amazon.com/elastictranscoder/

[15] AWS, “AWS Fargate - Run containers without having to manage servers or clusters”. Accessed: Nov. 22, 2023. [Online]. Available: https://aws.amazon.com/fargate/

[16] AWS, “AWS Lambda – Serverless Compute - Amazon Web Services”. Accessed: Nov. 22, 2023. [Online]. Available: https://aws.amazon.com/lambda/