**Assignment 3: Serverless/Event-driven Architectural Design Report**

*Duy Hung Do Hoang Tung Vu Duy Anh Do*

*Lab Section: Monday 1:00 PM Lab Section: Monday 1:00 PM Lab Section: Monday 1:00 PM*

*Swinburne University of Technology Swinburne University of Technology Swinburne University of Technology*

[*104182779@student.swin.edu.au*](mailto:104182779@student.swin.edu.au)[*104221522@student.swin.edu.au*](mailto:104221522@student.swin.edu.au)[*104653686@student.swin.edu.au*](mailto:104653686@student.swin.edu.au)

This report presents groundbreaking cloud architecture for a photo album application. To optimize data transfer and differentiation, I have decided to introduce a serverless architecture to replace the traditional relational database. Various AWS services have been utilized in this innovative approach, such as Amazon Recognition, Route 53, SNS, SQS, Amplify, API Gateway, etc. Our goal is to enhance the performance of photo and video uploads while minimizing response time. This report is detailed, covering specific use cases and thoroughly documenting data transformations. Additionally, it provides visual illustrations through diagrams, addressing special business requirements including cost-effectiveness and design criteria to deliver a groundbreaking and successful solution. We hope this approach will make the photo album application as flexible as possible.

Index terms—Cloud Computing, System Architecture, Serverless, Event-driven.

1. Introduction

In today's rapidly evolving digital landscape, technology serves as a catalyst for driving business success. Cloud computing stands out as a pivotal advancement, offering an effective avenue for business stakeholders and software developers to establish and expand their online presence. The concept of cloud computing has transformed the traditional infrastructure of businesses, offering a cost-effective and resource-efficient approach. As the boundaries between on-premise and cloud-based web applications continue to blur, our Photo Album application is poised to capitalize on the benefits of cloud technology through AWS (Amazon Web Services). With the support of AWS, managing our cloud application promises to be a seamless and hassle-free experience, ensuring optimal performance and scalability for our users.

One of the major advantages of AWS is its comprehensive suite of tools for code compilation, data collection, and application integration, all without the need to manage servers. Serverless architectures, which are powered by event-driven triggers, ensure the responsiveness and scalability of websites. By adopting a serverless approach, we can redirect our attention towards building the architecture itself without the burden of server management complexities. In the case of our Photo Album application, the primary focus lies in facilitating the seamless upload and sharing of captivating media, including images and videos. To achieve this objective, the proposal encompasses the core functionalities of the website.

* Upload files: There are different types of images (PNG, JPG, JPEG, etc…) and videos (GIF, MP4, etc..) should be resized and reprocessed to basic formats. And all of these files have to healthy file, no or a little bit virus or daring.
* Viewing files: We have to create a dashboard which will be the main page that I have pasted media files. Authorized users have the ability to search and refine their desired images or videos by utilizing metadata linked to the media, such as title, description, creation date, keywords, or AI-generated tags.

1. Architectural Diagram

The diagram below represents our description of the demo architecture showcasing the AWS services we propose to utilize. These AWS services are tightly integrated with each other through tiers such as front-end, login, authentication, and backend database tier. Additionally, in this exercise, we further expand the API processing tier, image recognition, and video differentiation layers.

A close-up of a grid

Description automatically generated

*Figure 1 Architectural Diagram*

1. Collaboration Diagrams

**1, Front-end tier**

In this particular tier dedicated to the front-end, a detailed explanation is provided regarding the architectural design of the Content Delivery Network (CDN) within the AWS infrastructure. In this context, the utilization of Route 53 is highlighted for the purpose of domain resolution, complemented by the inclusion of the Data Flow service and AWS CloudFront which are instrumental in the efficient distribution of content. An intricate approach has been devised which involves the creation of two distinct login paths that serve the dual purpose of enhancing security measures and optimizing the operational efficiency of each individual service. The initial pathway allows users to access the desktop server in order to conduct searches, following which the data is meticulously stored and subsequently transmitted to AWS Amplify by means of Route 53. On the other hand, the secondary method entails the deployment of a cloud-based streaming service exclusively tailored for handling content files. Through strategic utilization of AWS CloudFront, the CDN content is seamlessly delivered to multiple Edge locations. In instances where the CDN cache may not be readily accessible or encounters challenges related to data conversion, a proactive measure is in place where the nearest web server is prompted to facilitate access to the media content. The seamless integration of these two distinct methodologies has been meticulously orchestrated with the primary objective of guaranteeing swift retrieval speeds and ensuring a streamlined process for content distribution to users, ultimately culminating in an elevated and enriched overall user experience.

A diagram of a computer

Description automatically generated

*Figure 2 Front-end tier*

And of course, it's essential to incorporate a Cloud Monitoring model and an IAM role to enhance security and handle data cleanliness and dirtiness flexibly.

A grid with green objects

Description automatically generated with medium confidence

*Figure 3 Monitor and Role*

In the User Authentication step, I create a Lambda function to handle account registration or login. For registered accounts, the customer's data entered will be sent to the Cognito User Pool, where the registration account will be linked with Google, Twitter, Facebook, etc., depending on the user's choice. As for login accounts, the data will be forwarded to the Cognito Identity Pool to verify the account, then allow customers to access the website through the API gateway. This API is also connected to Amplify to reduce data access time and can create flexible and efficient backend APIs for the application. Additionally, the API can provide a certain number of endpoints for functions such as registration, login, data retrieval, etc.

A diagram of a software application

Description automatically generated

*Figure 4 User Authentication*

In the Successful Login tier, after accessing the API gateway, customers can input data into the website, which is then stored by the Service Categorizer by creating a new lambda and adding an S3 trigger. Afterwards, the website will send back confirmation of the data to be stored and forward it for categorization via SNS notifications. Here, SNS serves the dual role of notification and signaling for the data categorization process, handling both regular notifications and signaling to the data categorization component for images and videos after this step.

A diagram with a logo and arrows

Description automatically generated with medium confidence

*Figure 5 Login Tier*

In the data processing section, I divide it into two categories: image processing and video processing. For both parts, I use SQS to manage message queues from SNS between different components of the application. SQS plays a role in minimizing data loss and can help independent components operate independently. For each processing category, I create four different lambda functions for various functions such as data array creation, data processing, data transformation, and data completion. The difference between these steps lies in the transcoding or data conversion process. Since image data is much smaller than video data, in the video processing section, I utilize an elastic transcoder to handle large data more smoothly. In each processing section, I also implement data search functionality to help users understand and track the progress of image or video processing and enable them to eliminate unnecessary data at that moment.

A diagram of a software process

Description automatically generated

*Figure 6 Image and Video processing*

In the final section, I establish a comprehensive data storage solution using DynamoDB (RDS database), which is connected to phpMyAdmin for efficient data analysis of both good and bad data (S3 original and S3 processed). With RDS DynamoDB connected and receiving data from both image and video processing transcoders, RDS stores character and text-based data such as code languages to render images and videos via S3. This RDS service also shares data with the Search Categorizer to filter out unused data. The S3 bucket original stores image and video data that cannot be directly accessed for various reasons, and they are also connected to the Search Categorizer and the CreateThumbnail process. Meanwhile, the S3 bucket processed stores successfully processed image and video data from the Processed images and Process videos lambda functions. This section represents the output visible to users on the website.

A diagram of a data server

Description automatically generated

*Figure 7 Data tier*

1. Service Description

*1, S3 storage*

***Functionality***: is a cloud-based storage service provided by Amazon Web Services (AWS). Its primary functionality lies in providing scalable and durable object storage, allowing users to store and retrieve any amount of data securely over the internet. S3 is designed to be highly available and reliable, offering 99.999999999% (11 nines) durability and multiple redundancy options. It supports various storage classes to optimize cost and performance, along with features like versioning, lifecycle policies, and access controls for data management and security.

***Explanation of Use***: Amazon S3 is extensively used across industries and applications for a multitude of purposes. It serves as a central repository for storing and managing static and dynamic content, including images, videos, documents, backups, and logs. Organizations leverage S3 for data backup, archival, and disaster recovery, as well as for hosting static websites and distributing content globally through integration with AWS CloudFront. Additionally, S3 plays a crucial role in data lakes and analytics, acting as a scalable and cost-effective storage solution for big data processing, machine learning, and data warehousing. Overall, Amazon S3 provides a versatile and reliable platform for storing, managing, and accessing data in the cloud.

*2, Route 53*

***Functionality***: Route 53, the robust DNS web service offered by AWS, provides a comprehensive array of functionalities essential for the management of domain names and the routing of internet traffic. This particular service grants users the ability to efficiently register domain names, thereby aiding in the creation of unique website addresses. Furthermore, Route 53 operates as an immensely scalable DNS service, responsible for the translation of domain names into corresponding IP addresses to seamlessly guide users towards the appropriate web resources. In addition to this, it incorporates sophisticated traffic management capabilities, which encompass the utilization of health checks for monitoring the status of endpoints to ensure that traffic is exclusively directed towards resources that are in optimal condition. Moreover, Route 53 empowers users to deploy a variety of routing policies, including weighted routing, latency-based routing, and geolocation routing, all of which are designed to enhance the distribution of traffic across multiple endpoints. What's more, it also extends support for DNSSEC, a security enhancement that involves digitally signing DNS data to mitigate the possibility of potential attacks related to DNS.

***Explanation of Use***: Route 53 is widely utilized by businesses, organizations, and developers in practical terms to effectively handle their online presence and efficiently steer traffic towards their web resources. Through the utilization of Route 53, users have the ability to easily register domain names and smoothly guide traffic to their websites that are hosted on AWS or alternative platforms by configuring DNS records. Moreover, Route 53 seamlessly integrates with various AWS services such as Elastic Load Balancing (ELB), which empowers users to evenly distribute incoming traffic among multiple EC2 instances or containers to enhance scalability and reliability. Furthermore, organizations make use of Route 53 for the purpose of disaster recovery, establishing failover routing to guarantee uninterrupted service availability in the event of outages. In addition, businesses operating on a global scale take advantage of Route 53's sophisticated routing features, which include latency-based and geolocation-based routing, to direct users to the closest or most suitable resources based on their geographical location or network circumstances. In essence, Route 53 plays a pivotal role in the management of DNS and routing infrastructure, offering scalability, reliability, and advanced routing functionalities to cater to the diverse requirements of contemporary web applications and services.

*3, CloudFront*

***Functionality***: Amazon CloudFront is a content delivery network (CDN) service provided by AWS, offering fast and secure delivery of static and dynamic web content to users across the globe. CloudFront functions by caching content at edge locations strategically positioned worldwide, reducing latency and improving the overall performance of web applications. It supports various content types, including web pages, images, videos, and APIs, and offers advanced caching capabilities to optimize content delivery. Additionally, CloudFront integrates seamlessly with other AWS services, enabling developers to leverage features such as custom SSL certificates, access control, and real-time logging for enhanced security and monitoring.

***Explanation of Use***: CloudFront is widely utilized by businesses and developers to accelerate the delivery of their web content to end-users, resulting in improved user experience and higher engagement. By leveraging CloudFront, organizations can distribute content from their origin servers (such as Amazon S3 buckets, EC2 instances, or custom HTTP servers) to edge locations worldwide, reducing the latency experienced by users accessing the content. This is particularly beneficial for serving large media files, streaming video content, or delivering dynamic web applications with low latency. Moreover, CloudFront enables users to enforce access policies, restrict unauthorized access to content, and mitigate distributed denial-of-service (DDoS) attacks through integration with AWS Web Application Firewall (WAF) and AWS Shield. Overall, CloudFront serves as a scalable and reliable CDN solution for optimizing content delivery, enhancing performance, and ensuring global availability of web applications and services.

*4, Amplify*

***Functionality***: AWS Amplify is a comprehensive development platform provided by Amazon Web Services, with the aim of simplifying the process of constructing and deploying scalable and secure full-stack applications. Amplify offers a range of tools and services that make common development tasks easier, such as authentication, data storage, APIs, analytics, and hosting. Developers can utilize Amplify to efficiently create and oversee backend resources, connect with frontend frameworks, and automate deployment processes. Supporting well-known frontend frameworks like React, Angular, and Vue.js, along with native mobile platforms, Amplify empowers developers to concentrate on developing innovative features and providing value to users, eliminating the necessity for intricate infrastructure setup or management.

***Explanation of Use***: AWS Amplify is extensively utilized by developers and development teams to hasten the process of developing and deploying web and mobile applications. Through the utilization of Amplify, developers can effortlessly incorporate authentication, data storage, and API functionality into their applications, consequently reducing the time and effort required for development. Amplify delivers a cohesive development environment encompassing both frontend and backend components, enabling developers to smoothly integrate AWS services like Amazon Cognito for user authentication, Amazon DynamoDB for data storage, and AWS Lambda for serverless compute. Furthermore, Amplify provides functionalities such as real-time analytics, offline data synchronization, and continuous deployment, empowering developers to construct resilient and scalable applications with ease. In general, Amplify functions as a robust and adaptable platform for contemporary application development, allowing developers to concentrate on creating exceptional user experiences while AWS manages the foundational infrastructure and services.

*5, API gateway*

***Functionality***: AWS API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. It acts as a gateway for applications to connect with backend services hosted on AWS or outside of AWS. API Gateway supports RESTful and WebSocket APIs, allowing developers to build flexible and scalable architectures. It provides features such as request and response transformation, rate limiting, authentication and authorization, caching, logging, and monitoring, enabling developers to customize and control the behavior of their APIs. Additionally, API Gateway integrates seamlessly with other AWS services like Lambda, DynamoDB, and S3, making it a central component for building serverless and microservices-based applications.

***Explanation of Use***: AWS API Gateway is used by developers and organizations to expose backend services and functionalities through well-defined APIs. It serves as a front door for applications to access backend resources securely and efficiently. Developers can use API Gateway to create APIs for web and mobile applications, IoT devices, and third-party integrations, enabling seamless communication between clients and backend services. API Gateway simplifies the process of API development and management by providing a variety of features and tools to handle common tasks such as request validation, authorization, and traffic management. Organizations leverage API Gateway to build scalable and reliable APIs, reduce development time and costs, and improve overall agility and innovation. Additionally, API Gateway plays a crucial role in enabling serverless architectures and microservices patterns, allowing developers to focus on building core business logic while AWS manages the underlying infrastructure and scalability concerns.

*6, Lambda*

***Functionality***: AWS Lambda is a service for serverless computing that enables developers to execute code without the need to provision or manage servers. Developers can simply upload their code as functions, and AWS takes care of deploying, scaling, and maintaining the necessary infrastructure. Lambda supports various programming languages like Node.js, Python, Java, and Go, allowing developers to create functions in their preferred language. These functions are activated by different events such as HTTP requests, data changes in AWS services like S3 or DynamoDB, or custom events. Lambda functions operate in a stateless manner, ensuring that each function invocation is independent of previous ones, which facilitates the creation of highly scalable and event-driven architectures.

***Explanation of Use***: Developers use Lambda to run code in response to events, eliminating the need to handle servers and infrastructure. Lambda functions serve various purposes, such as data processing, file processing, real-time stream processing, image and video processing, IoT data processing, and more. Organizations employ Lambda to develop scalable and cost-efficient solutions, paying only for the compute time their functions use, without incurring charges when functions are inactive. Lambda functions seamlessly integrate with different AWS services like API Gateway, S3, DynamoDB, SNS, and SQS, enabling developers to easily construct complex applications. Moreover, Lambda allows for quick development and deployment cycles, enabling developers to concentrate on coding and delivering value to users without concerns about managing infrastructure.

*7, SNS*

***Functionality***: Amazon Simple Notification Service (SNS) is a fully managed messaging service provided by AWS that enables developers to send messages or notifications to distributed systems, microservices, mobile devices, and other endpoints. SNS allows users to decouple message producers from consumers, ensuring reliable and scalable message delivery. Users can publish messages to topics, which act as communication channels, and subscribers can receive notifications from these topics based on their preferences. SNS supports multiple protocols, including HTTP, HTTPS, email, SMS, SQS, Lambda, and mobile push notifications, providing flexibility in message delivery across different platforms and endpoints. Additionally, SNS offers features such as message filtering, message attributes, message encryption, and message retries to ensure message delivery reliability and security.

***Explanation of Use***: AWS SNS is commonly used by developers and organizations to build event-driven architectures, push notifications, and communication systems. Developers leverage SNS to send notifications to users about important events, such as order confirmations, system alerts, status updates, and promotional messages. SNS can be integrated with various AWS services, such as Lambda, SQS, and CloudWatch, allowing developers to trigger actions or workflows in response to messages. Organizations use SNS to build scalable and resilient systems, as SNS automatically handles message delivery, retries, and error handling. With SNS, developers can easily send messages to multiple subscribers or endpoints simultaneously, enabling efficient communication and collaboration across distributed systems. Additionally, SNS supports message filtering based on attributes or message content, allowing subscribers to receive only relevant notifications, reducing message overhead and improving efficiency.

*8, SQS*

***Functionality***: Amazon Simple Queue Service (SQS) is a message queuing service managed by AWS, designed to decouple and scale microservices, distributed systems, and serverless applications. Users can use SQS to send, store, and receive messages between various software components or microservices, enabling reliable and asynchronous communication. Messages are temporarily stored in queues until they are processed by consumers. SQS offers two queue types: standard queues and FIFO (First-In-First-Out) queues. Standard queues prioritize ordering and ensure message delivery at least once, while FIFO queues guarantee message sequencing and processing exactly once. SQS automatically adjusts to demand, manages message retries, and supports customizable message retention periods and visibility timeouts.

***Explanation of Use***: AWS SQS is widely utilized in distributed systems, microservices architectures, and event-driven applications to separate components, enhance scalability, and ensure fault tolerance. Developers utilize SQS to establish asynchronous communication patterns, allowing producers and consumers to function independently without direct interaction. SQS facilitates dependable message delivery across various systems or services, even in the presence of failures or high traffic volumes. Applications make use of SQS to execute workflows, handle background tasks, oversee job queues, and synchronize workloads across distributed environments. Moreover, SQS seamlessly integrates with other AWS services like Lambda, SNS, and EC2, empowering developers to construct scalable and resilient applications effortlessly. Through the utilization of SQS, developers can boost application performance, diminish coupling among components, and enhance overall system reliability and scalability.

*9, Recognition*

***Functionality***: Amazon Rekognition is a service provided by AWS that utilizes deep learning for analyzing images and videos. It empowers developers to extract valuable insights from visual content through advanced machine learning algorithms. Users can leverage Rekognition for tasks like object detection, facial recognition, text detection, scene comprehension, celebrity identification, and content moderation. This service excels in accurately identifying objects, people, text, and activities in images and videos. By offering APIs for seamless integration into applications, Rekognition enables developers to create intelligent systems capable of understanding and interpreting visual data.

***Explanation of Use***: Amazon Rekognition is extensively used in various industries and applications for various purposes. Content moderation systems commonly employ it to identify and remove inappropriate or harmful content from user-generated media. E-commerce platforms use Rekognition for product recognition and visual search, allowing users to search for products using images rather than text. In the entertainment sector, Rekognition drives recommendation engines through analyzing user preferences and content attributes. Law enforcement agencies use Rekognition for facial recognition to spot suspects in surveillance footage or social media images. Moreover, Rekognition finds applications in healthcare for medical image analysis, manufacturing for quality inspection, and agriculture for crop monitoring. In essence, Amazon Rekognition empowers developers to enhance their applications with robust image and video analysis capabilities, resulting in improved user experiences, operational efficiency, and decision-making processes.

*10, Cognito*

***Functionality***: With Cognito, developers can easily integrate user authentication mechanisms into their applications without the need to manage infrastructure or handle sensitive user data directly. The service provides secure storage of user credentials, supports multiple identity providers (such as Amazon, Google, Facebook, and Apple), and offers built-in support for multi-factor authentication (MFA). Additionally, Cognito synchronizes user data across devices, allowing seamless user experiences across platforms.

***Explanation of Use***: Amazon Cognito is commonly used by developers to add user authentication and access control features to their applications quickly and securely. It is particularly beneficial for web and mobile applications that require user registration, login, and personalized experiences. Developers can integrate Cognito into their applications using its SDKs and APIs, which provide easy-to-use methods for user authentication and authorization. Cognito's user pools allow developers to manage user accounts, groups, and permissions efficiently, while its identity pools enable seamless integration with external identity providers and federated identity management. Applications across various industries, including e-commerce, finance, healthcare, and gaming, utilize Cognito to enhance security, protect user data, and deliver personalized experiences to their users. Overall, Amazon Cognito simplifies the implementation of user authentication and authorization mechanisms, enabling developers to focus on building core application functionality while ensuring a secure and seamless user experience.

*11, IAM*

***Functionality***: AWS Identity and Access Management (IAM) is a web service that enables users to securely control access to AWS resources. IAM allows users to manage user identities, permissions, and policies within their AWS environment. With IAM, users can create and manage AWS users and groups, assign granular permissions to resources, and set up multi-factor authentication (MFA) for enhanced security. IAM also provides features such as role-based access control (RBAC), identity federation, and integration with AWS services for centralized access management.

***Explanation of Use***: AWS IAM is widely used by organizations to enforce security best practices and ensure compliance with regulatory requirements in their AWS environments. Administrators can use IAM to create and manage user accounts for employees, assign specific permissions to each user or group based on their roles or responsibilities, and enforce least privilege access policies to mitigate the risk of unauthorized access. IAM enables organizations to implement fine-grained access controls, restrict access to sensitive resources, and monitor user activity through comprehensive logging and auditing capabilities. Additionally, IAM integrates seamlessly with other AWS services, allowing users to control access to various AWS resources, including EC2 instances, S3 buckets, RDS databases, and more. Overall, AWS IAM provides a robust and scalable identity and access management solution for securing AWS resources and protecting sensitive data in the cloud.

*12, CloudWatch*

***Functionality***: AWS CloudWatch is a monitoring and observability service provided by Amazon Web Services (AWS). It collects and tracks metrics, logs, and events from various AWS resources and applications, allowing users to gain insights into their system's performance, operational health, and resource utilization. CloudWatch offers real-time monitoring capabilities, enabling users to visualize and analyze metrics, set alarms, and automate actions based on predefined thresholds or anomalies. Additionally, CloudWatch provides centralized logging and monitoring solutions, allowing users to aggregate, search, and analyze log data from multiple sources for troubleshooting and diagnostics.

***Explanation of Use***: AWS CloudWatch is commonly used by AWS customers to monitor the performance and health of their cloud infrastructure and applications. Users can leverage CloudWatch to collect and visualize metrics such as CPU utilization, network traffic, and disk usage, enabling them to identify performance bottlenecks, optimize resource allocation, and ensure cost efficiency. CloudWatch alarms can be set up to notify users of any deviations from expected performance metrics or to trigger automated responses, such as scaling instances or sending notifications to stakeholders. Moreover, CloudWatch Logs enables users to centralize and analyze log data generated by AWS services and applications, facilitating troubleshooting, compliance, and security monitoring efforts. Overall, AWS CloudWatch provides essential monitoring and observability capabilities to help users maintain the reliability, availability, and performance of their AWS environments.

*13, RDS database (DynamoDB)*

***Functionality***: (Werner, 2012)DynamoDB supports both document and key-value data models, allowing users to store and retrieve structured and semi-structured data with low latency. It offers features such as automatic data replication across multiple Availability Zones, built-in security controls, and flexible indexing options. DynamoDB also provides advanced capabilities like transactions, conditional writes, and on-demand capacity scaling, enabling users to build highly responsive and resilient applications.

***Explanation of Use***: DynamoDB is widely used by developers and organizations to build fast, scalable, and reliable applications that require low-latency access to large volumes of data. It is particularly well-suited for use cases such as web and mobile applications, gaming, ad tech, IoT, and real-time analytics. Developers can use DynamoDB to store and retrieve user profiles, session data, product catalogs, gaming leaderboards, and more. With its flexible data model and support for complex queries, DynamoDB enables users to efficiently handle diverse data types and access patterns. Additionally, DynamoDB integrates seamlessly with other AWS services like Lambda, API Gateway, and S3, allowing users to build fully serverless architectures and streamline their development workflows. Overall, DynamoDB provides a reliable and scalable database solution for modern cloud applications.

*14, Label search*

***Functionality***: Label search is a function offered by different AI and machine learning platforms like Amazon Rekognition. It allows users to search for images or videos by identified labels, objects, or scenes in the media. This feature utilizes computer vision algorithms to analyze visual content and create metadata tags for objects, concepts, or activities in the media files. By using these labels, users can easily search for relevant media assets without having to manually check each file.

***Explanation of Use***: Label search is widely used in media management, content moderation, and search applications where there is a need to efficiently organize, index, and retrieve large volumes of images or videos. For instance, e-commerce platforms can utilize label search to categorize product images according to attributes like color, style, or category, making it easier for shoppers to locate items quickly through visual search. Likewise, media libraries, digital asset management systems, and social media platforms use label search to empower users to filter and explore content based on specific topics, themes, or objects. By integrating label search functionality into their applications, developers can enhance user experiences, streamline content discovery, and enhance the overall usability of their platforms.

V, Requirements met

*1, Minimize the need for in-house systems administrations*

In this design, we have minimized the number of services to ensure the requirement for in-house systems administration is kept to a minimum, while also optimizing operational efficiency through various login pathways to reduce administrative burdens.

S3 storage addresses the entire media storage requirement to minimize in-house systems administration. S3 boasts immense and secure data storage capabilities, eliminating the need for dedicated system administration services. Particularly, S3's high level of automation scales various features of the service, enabling the company to focus less on this aspect and attend to other critical matters such as user experience testing for app functionality.

And once we acquire S3 storage, DynamoDB becomes essential. It is a NoSQL database that functions independently of EC2 instances, eliminating the need for manual involvement in managing the database. Like S3, DynamoDB possesses automatic scaling features that adapt to varying traffic patterns in real-time, guaranteeing peak performance under heavy loads without the need for continuous supervision. This improves operational effectiveness and notably lessens the burden on in-house management.

API Gateway plays a pivotal role in managing APIs effectively, offering a fully managed solution for maintenance, security, and monitoring tasks. By utilizing API Gateway, businesses can delegate the complexities of API management to a managed service, thereby reducing the need for extensive system administration. This strategic approach enables organizations to allocate resources more efficiently towards core application development, rather than dedicating them to infrastructure upkeep.

A diagram of a network

Description automatically generated

*Figure 8 API Gateway Design*

*2, Meeting customer demands and updating every six months*

To effectively manage the doubling of demand every six months, the architectural design strategically incorporates Lambda functions as a key component to handle fluctuating workloads.

We have developed numerous Lambda functions to improve data processing. Given the statistics indicating a doubling of users every 6 months, I depend on Lambda's automatic capability. These functions adapt dynamically to gather data, process it autonomously, and optimize resource usage. By avoiding EC2 instances, this automated system efficiently manages data requests without manual involvement, guaranteeing top performance.

Furthermore, DynamoDB improves the efficiency of storing data by utilizing the Auto Scaling service to adapt provisioned throughput capacity according to current traffic patterns. This approach aids in avoiding throttling and maintaining peak productivity when facing unexpected surges in demand. Additionally, DynamoDB offers auto-scaling functionalities to sustain performance levels when adjusting read and write speeds. By integrating with Lambda, DynamoDB establishes a strong base for managing extensive datasets.

API Gateway smoothly enhances Lambda's abilities by flexibly scaling its capability to manage growing API traffic. When the user base grows, API Gateway adapts automatically to handle different levels of traffic, guaranteeing a seamless and scalable connection between frontend and backend services. This adaptive scaling feature boosts system resilience, enabling it to effectively manage sudden surges in traffic without compromising overall reliability. API Gateway's effective handling of API workloads aligns with the primary objective of meeting increasing demand.

*3, Adopt a serverless/event-driven solution*

Opting for a serverless and event-driven solution in architectural design represents a strategic decision aimed at boosting efficiency, agility, and resource optimization. At the heart of this approach lies the hosting of static websites in an S3 bucket and the implementation of backend logic through Lambda functions.

With Automatic Scaling integration, Lambda plays a crucial role in achieving relative efficiency due to the absence of EC2 instances. This architecture doesn't require all services to run 24/7, activating only when needed. Therefore, the usage costs significantly decrease as there's no need for elastic IP to maintain data effectively, allowing for more flexible storage. These Lambda functions scale smoothly as workload increases or decreases, optimizing resource utilization.

The event-driven nature of Lambda is fundamental and essential, as it perfectly aligns with the company's inclination towards a system that is highly responsive to various stimuli and triggers. Lambda showcases its remarkable capability to promptly react the moment specific triggers are introduced, enabling seamless data processing to take place within a predetermined and fixed timeframe. This event-driven architecture not only boosts the system's adaptability and agility but also ensures that it remains adept at swiftly adjusting to evolving circumstances while effectively managing a diverse array of tasks that are initiated by particular events.

Another advantage is the cost-effectiveness that comes from not using EC2 instances. In addition to EC2 services, there are charges for keeping elastic IPs when they are created, and they get lost if the elastic IPs are not launched via a launch template. The combination of Lambda and S3 triggers offers flexibility as the user numbers vary, resulting in fluctuations in data volume without compromising the efficiency of the system. This is in line with the company's objectives of being cost-effective and scalable.

*4, Improve the response time for global except Australia*

The decision to substitute the conventional relational database with DynamoDB is in harmony with the company's goal of attaining a quicker and more economical data storage solution. This move towards DynamoDB is driven by its array of Key Features aimed at boosting performance. DynamoDB's key-value data model, coupled with automatic scaling capabilities and low-latency performance, positions it as a viable substitute for traditional relational databases. The key-value model not only facilitates the efficient storage and retrieval of basic metadata associated with photos and videos but also contributes to an enhanced performance. Meanwhile, the automatic scaling feature plays a crucial role in ensuring that the database dynamically adjusts its capacity in real-time, reacting to fluctuating traffic patterns and averting any potential performance bottlenecks that may arise during periods of heightened demand. This level of flexibility ultimately leads to a more agile, responsive, and efficient data storage solution overall.

DynamoDB boasts a multitude of key features that enhance its functionality, such as the pay-as-you-go pricing model which guarantees cost-effectiveness, especially beneficial for applications with fluctuating data storage needs. In contrast to rigid-schema relational databases, DynamoDB empowers the application to dynamically adjust its throughput capacity in accordance with demand, thereby optimizing expenses during periods of reduced activity. This strategic approach resonates with the company’s aim of realizing a financially efficient solution without compromising on performance.

*5, Explore more cost-effective options in a new AWS insfrastructure*

Addressing the challenge of slow global response times is an essential and pivotal aspect that must be carefully considered and meticulously incorporated into the overarching framework of architectural design, as it significantly impacts the overall performance and user experience of any digital infrastructure. Amazon Cloud Front, an advanced and innovative content delivery network service provided by the tech giant Amazon, emerges as a crucial player and influential force in optimizing and enhancing response times on a global scale, thus playing a central and essential role in achieving swift and efficient delivery of digital content to end-users across various geographical locations.

Here, the cost will heavily depend on CloudFront as it serves as the storage location at Edge Locations worldwide, including cached data from static web files and user-uploaded content. This helps reduce response latency for users outside Australia. Edge Locations, when combined with S3, ensure reliability in storage and global accessibility. This trend is currently favored due to the strategic use of content delivery networks aligning with the company's global goals.

CloudFront's capability to deliver content from edge locations closest to users ensures expedited delivery times and enhanced webpage loading speeds, ultimately improving the overall user experience. Through seamless integration with S3, content can be dynamically processed and customized, optimizing the delivery process further. This approach guarantees minimal latency for users accessing the application, irrespective of their geographical location, thereby contributing to enhanced global response times.

*6, Handle video media*

Integrating Elemental MediaConvert into the architecture addresses the requirement for efficient video media processing.

Handling video media in AWS involves leveraging a combination of powerful services tailored to efficiently store, process, and deliver video content. At the core of this architecture is Amazon S3 (Simple Storage Service), which provides highly scalable, durable, and cost-effective storage for video files. With S3, video files can be securely uploaded to designated buckets, making them accessible for processing and distribution. To ensure optimal playback across various devices and platforms, AWS Elemental MediaConvert comes into play. In this situation, I choose Elastic Transcoder instead of Elemental. This file-based video transcoding service allows for seamless conversion of media files into different formats, bitrates, and resolutions. Furthermore, Amazon CloudFront, a content delivery network (CDN) service, accelerates the delivery of video content by caching it at edge locations closer to viewers, thereby reducing latency and enhancing streaming performance. Additionally, Amazon Elastic Transcoder offers scalable media transcoding capabilities, automatically converting video files stored in S3 into formats optimized for playback on different devices. AWS Elemental MediaPackage securely packages and delivers live or on-demand video content, supporting features like adaptive bitrate streaming and content encryption. For advanced video analysis and enrichment, Amazon Rekognition Video employs deep learning algorithms to extract metadata, detect objects and faces, recognize text, and perform content moderation. By harnessing the capabilities of these AWS services, organizations can build robust, scalable, and feature-rich solutions to effectively handle video media in their applications.

*7, The media can be uploaded by users in all sorts of formats.*

The utilization of a sophisticated combination of AWS services, such as Amazon S3, Simple Notification Service (SNS), Simple Queue Service (SQS), and AWS Lambda functions, within the architecture enables the dynamic processing of uploaded media. Through the strategic utilization of SNS and SQS in a decoupled fashion, the system guarantees extensibility and flexibility in managing a variety of media processing tasks effectively. SNS functions as a robust messaging service, disseminating notifications across multiple SQS queues, while SQS facilitates the simultaneous processing of diverse tasks associated with media uploads. This well-structured and decoupled approach not only fosters scalability and resilience but also enables each processing task to operate independently, thus facilitating seamless expansion to accommodate new processing requirements as they arise.

Upon a user initiating the process of uploading media files into the specified S3 bucket, a series of S3 Event Notifications are triggered seamlessly, setting off a chain reaction of events meticulously orchestrated through the central SNS topic. This meticulously designed system then kick-starts downstream processing tasks, each intricately organized and managed through individual SQS queues, meticulously tailored to handle a specific type of media processing task with precision and efficiency. The Lambda functions, intricately linked to these SQS queues, operate autonomously to carry out the designated tasks in parallel, guaranteeing a streamlined workflow that minimizes the risk of bottlenecks occurring during the processing phase. This innovative "fan-out" architecture plays a pivotal role in ensuring the swift and effective processing of both photos and videos, adeptly catering to a diverse range of processing requirements while upholding the system's scalability and reliability standards. Furthermore, the inherently decoupled nature of this design not only bolsters fault tolerance but also simplifies the process of load balancing, thereby guaranteeing uninterrupted operation even during peak periods of high demand.

A diagram of a diagram

Description automatically generated

*Figure 9 Fan-out approach of developing*

*8, Conclusion*

In summary, the services I utilize all contribute to high productivity without incurring unnecessary costs, ensuring reliability and security for the image and video storage application, and meeting the company's development needs. It aligns with the company's business goals of minimizing or completely eliminating server usage.

VI, Description and justification

*1, Performance*

***CloudFront***: CloudFront is a content delivery network (CDN) service provided by Amazon Web Services (AWS) that caches content globally, thereby enhancing response times for users across the globe. By caching content at edge locations closer to users, CloudFront reduces latency and delivers content more quickly, resulting in a smoother and faster user experience. This distributed caching strategy ensures that users accessing the application from different geographical locations can enjoy optimized performance regardless of their proximity to the application's origin server.

***Lambda***: Lambda is a serverless compute service offered by AWS that enables automatic scaling and event-driven execution of code in response to triggers or events. With Lambda, developers can write functions that respond to various events, such as HTTP requests, file uploads, or database updates, without having to provision or manage servers. This serverless architecture allows Lambda functions to scale automatically in response to changes in workload, ensuring seamless handling of peak demands, such as media uploads and processing. By eliminating the need to manage infrastructure, Lambda simplifies the deployment and operation of applications, allowing developers to focus on writing code and delivering value to customers.

***API Gateway***: API Gateway is a fully managed service provided by AWS that allows developers to create, publish, monitor, and secure APIs at any scale. With API Gateway, developers can efficiently manage incoming API requests, automatically scale to handle varying traffic levels, and distribute requests across multiple backend services. This helps improve the performance, reliability, and security of APIs, ensuring that applications can seamlessly interact with backend services and external systems. Additionally, API Gateway provides features such as authentication, authorization, and throttling to protect APIs from abuse and ensure compliance with security and regulatory requirements.

***Simple Storage Service (S3)***: Amazon S3 is a highly scalable, reliable, and cost-effective object storage service provided by AWS. S3 is designed to handle massive amounts of data and concurrent requests from users around the world, making it ideal for storing and serving static assets such as images, videos, and documents. By leveraging S3, developers can offload the storage and delivery of content to AWS, ensuring optimized performance and reduced latency for end users. S3 integrates seamlessly with other AWS services, such as CloudFront, enabling developers to build scalable and reliable applications with ease.

***DynamoDB***: DynamoDB is a fully managed NoSQL database service provided by AWS that offers seamless scalability, high performance, and low latency for applications with varying data storage requirements. Unlike traditional relational databases, DynamoDB uses a simple key-value data model and supports horizontal scaling, allowing it to seamlessly scale as storage demand grows. This makes DynamoDB an ideal choice for applications that require fast and predictable performance, such as those with large-scale, high-traffic workloads. Additionally, DynamoDB offers built-in security features, automatic backups, and multi-region replication, ensuring data durability and availability.

***Cognito***: Amazon Cognito is a fully managed authentication and user management service provided by AWS that scales to support millions of users and authenticates users through multiple identity providers, such as social identity providers, enterprise identity providers, and SAML-based identity providers. With Cognito, developers can easily add user sign-up, sign-in, and access control to their applications, reducing the complexity and overhead of managing user identities and credentials. Cognito integrates seamlessly with other AWS services, such as API Gateway and Lambda, enabling developers to build secure and scalable applications with ease.

***Rekognition***: Amazon Rekognition is a fully managed computer vision service provided by AWS that enables developers to analyze images and videos to identify objects, people, text, scenes, and activities. With Rekognition, developers can build applications that automatically tag and categorize images, extract text from images, and detect and recognize faces in photos and videos. By leveraging machine learning algorithms and deep learning models, Rekognition delivers accurate and reliable results, making it ideal for a wide range of use cases, including content moderation, image recognition, and video analysis.

***Simple Queue Service (SQS)***: Amazon SQS is a fully managed message queuing service provided by AWS that decouples the components of a distributed application and prevents overload and job loss by buffering messages between components. With SQS, developers can build scalable and reliable distributed systems that can handle large volumes of messages and process them asynchronously. SQS offers two types of message queues: standard queues and FIFO (First-In-First-Out) queues, each with its own set of features and capabilities. By decoupling components and buffering messages, SQS helps improve the reliability, scalability, and performance of distributed systems, ensuring seamless operation even under high loads and peak demand.

***Simple Notification Service (SNS)***: Amazon SNS is a fully managed pub/sub messaging service provided by AWS that enables developers to build distributed systems and microservices that can communicate with each other asynchronously. With SNS, developers can create topics and publish messages to those topics, which are then delivered to subscribers via various protocols, such as HTTP/HTTPS, email, SMS, and SQS. SNS decouples the components of a distributed system and enables parallel processing to improve performance, creating a fan-out architecture for extensibility. By providing reliable and scalable messaging capabilities, SNS helps developers build robust and scalable applications that can handle a wide range of use cases and scenarios.

*2, Reliability*

***Route 53***: Amazon Route 53 is a highly scalable and reliable Domain Name System (DNS) web service designed to route end users to internet applications by translating domain names into numeric IP addresses. It provides a globally distributed network of DNS servers strategically positioned around the world to ensure fast and consistent routing using custom domain names. With Route 53, businesses can manage domain registration, DNS routing, and health checking with ease, ensuring their web applications remain accessible and responsive to users worldwide.

***API Gateway***: Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. It efficiently handles high volumes of incoming API requests without compromising performance, incorporating features like API throttling and authentication to ensure a secure and reliable user experience. API Gateway acts as a front door for applications to access data, business logic, or functionality from backend services, abstracting away the complexity of managing infrastructure and allowing developers to focus on building innovative applications.

***S3***: Amazon Simple Storage Service (S3) is a scalable object storage service designed to store and retrieve any amount of data from anywhere on the web. It provides vast storage capacity and supports various media formats, making it ideal for securely storing uploaded photos and videos. With S3, businesses can benefit from durable, highly available, and cost-effective storage options, ensuring their data remains safe and accessible at all times. Additionally, S3 integrates seamlessly with other AWS services, enabling businesses to build scalable and reliable applications without the need for extensive infrastructure management.

***DynamoDB***: Amazon DynamoDB is a fully managed NoSQL database service designed for applications that require consistent, single-digit millisecond latency at any scale. It offers fast and reliable storage for metadata with a simple key-value data model and horizontal scaling capabilities. DynamoDB automatically scales up or down based on workload demands, ensuring optimal performance and cost efficiency. With its flexible data model and built-in security features, DynamoDB is an ideal choice for storing and retrieving structured data for a wide range of use cases, from gaming and mobile apps to IoT devices and real-time analytics.

*3, Security*

***Cognito***: Within the realm of AWS services, Cognito stands out as a highly robust user authentication management solution that plays a pivotal role in enabling the implementation of stringent security protocols. These protocols include but are not limited to token-based authentication, access control management, and robust data encryption mechanisms that are crucial for safeguarding sensitive information within the AWS environment. Furthermore, Cognito is designed to extend support for Multi-Factor Authentication (MFA), a key feature that significantly enhances the overall security posture by adding an extra layer of protection to user accounts.

***Lambda***: Among the array of security features offered by Lambda, one can find a comprehensive set of robust measures aimed at fortifying the overall security framework. These measures encompass the utilization of IAM roles, which facilitate granular access control, along with robust data encryption protocols that are instrumental in upholding data privacy standards within the AWS ecosystem. By implementing these security measures, Lambda ensures that data processed through its functions remains secure and protected from unauthorized access or breaches.

***S3***: In the domain of cloud storage services, S3 is recognized for its implementation of formidable security measures such as access control lists (ACLs), bucket policies, and server-side encryption protocols. These measures collectively work towards fortifying the security architecture of S3, thereby establishing a robust defense mechanism to protect stored data from potential security threats or unauthorized intrusions. Through the integration of these security features, S3 effectively safeguards the confidentiality and integrity of data stored within its infrastructure.

***DynamoDB***: As a prominent NoSQL database service, DynamoDB prioritizes the security of data both at rest and in transit by employing sophisticated encryption protocols. Additionally, DynamoDB enforces IAM policies and implements fine-grained access control mechanisms to regulate and restrict access to database tables and associated data. By incorporating these security measures, DynamoDB ensures that sensitive information is shielded from unauthorized access or malicious activities, thereby maintaining the integrity and confidentiality of data stored within its databases.

***API Gateway***: Within the context of API management, API Gateway emerges as a crucial component that leverages various security measures to authenticate and authorize requests effectively. These measures include the utilization of IAM roles, implementation of unique authentication methods, and integration with AWS Cognito for seamless request authentication and authorization processes. By leveraging these security features, API Gateway adeptly controls and restricts access to APIs based on user identities and permissions, thereby fortifying the overall security posture of API endpoints and ensuring a secure interaction environment for users.

Overall, the amalgamation of these selected AWS services showcases a sophisticated blend of high performance, scalability, reliability, and robust security features that align seamlessly with the design requirements of the Photo Album application. The cohesive integration of these services contributes significantly to the establishment of an architectural framework that is adept at efficiently managing dynamic workloads, preserving data integrity, and delivering a secure and seamless user experience within the AWS ecosystem.

*4, Cost*

A screenshot of a computer screen

Description automatically generated

*Figure 10 Cost use for this infrastructure*

VII, UML Diagram

A diagram of a computer program

Description automatically generated

*Figure 11 UML Diagram*

When a user decides to start the process of uploading media, the initial step involves the utilization of Route 53 and CloudFront to guide the media towards AWS Amplify. Within the confines of AWS Amplify, an upload request is created through an API, which triggers the commencement of the PUT method. Subsequently, the media is directed through the API to engage with the Service Categorizer function, responsible for allocating a specific category based on the type of media, whether it pertains to images or video content. These designated categories serve as informative signals for AWS SNS regarding the planned destination for the media content. Following this, AWS SNS takes charge and dispatches the media to AWS SQS for the purpose of queuing and subsequent processing.

Throughout this particular phase, both the original media file and its corresponding thumbnail undergo the process of being tagged using AWS Rekognition, after which they are stored within an S3 bucket with the assistance of the thumbnail's formation process. Simultaneously, the media content experiences transcoding through AET (Automatic Encoding Transcoder), leading to the creation of metadata. The transcoded media content is then relocated back into S3, whereas the related metadata is securely preserved within the confines of the DynamoDB database.

VIII, Alternative solutions

*1, EC2 instances and CloudFront, Edge Location*

The main difference between using EC2 instances combined with VPC and utilizing CloudFront and Edge Location along with AWS Amplify lies in the flexibility, scalability, and performance.

*a, Flexibility and Scalability:*

- ***CloudFront and Edge Location***: These services provide a globally distributed content delivery network with edge servers deployed in multiple locations worldwide. This enhances access speed and reduces latency for users everywhere. CloudFront also automatically scales to handle large traffic volumes without manual intervention.

- ***EC2 and VPC***: While EC2 instances offer a flexible environment for deploying applications and services, managing and scaling them requires more effort. You need to manually handle scaling and maintenance of EC2 instances as well as manage VPC resources like networking, security, and IP addresses.

*b, Performance*:

- ***CloudFront and Edge Location***: Optimized to deliver content at the nearest point to users, reducing latency and improving access speed. Edge Locations are optimized for both static and dynamic content delivery, enhancing overall application performance.

- ***EC2 and VPC***: Although EC2 instances can provide high performance, managing and optimizing performance for content delivery applications requires deep knowledge of system administration and networking.

Reasons not to use EC2 and VPC as a replacement for CloudFront and Edge Location along with AWS Amplify include:

- ***Complexity and Cost***: Managing and maintaining a network of EC2 instances and VPC can incur higher costs and complexity compared to using AWS managed services like CloudFront and Amplify.

- ***Reliability and Performance***: CloudFront and Edge Location are optimized for global content delivery and provide better performance compared to self-deployed EC2 servers.

*2, AWS fargate and AWS lambda*

When opting for AWS Lambda over AWS Fargate, there are numerous advantages to consider in terms of trigger integration, optimization, scalability, and cost-effectiveness:

*a. Trigger Integration*:

- ***AWS Lambda***: One of the key benefits of using Lambda is its ability to be triggered by a wide array of AWS services, including S3 events, API Gateway requests, SNS notifications, DynamoDB streams, and more. This seamless integration facilitates the creation of event-driven architectures, where actions are automatically initiated in response to specific events or system changes. The flexibility and versatility of Lambda's trigger capabilities enable developers to build robust and responsive applications.

- ***AWS Fargate***: While Fargate also allows integration with various AWS services, its trigger options may not be as extensive as those of Lambda. Typically, Fargate is utilized for running containerized applications within ECS or EKS clusters, with triggers relying on other AWS services or custom solutions. This may require additional setup and configuration compared to the straightforward trigger integration offered by Lambda.

*b. Optimization*:

- ***AWS Lambda***: Lambda functions are entirely managed by AWS, relieving developers of the burden of infrastructure provisioning, scaling, and maintenance. The automatic scaling of Lambda based on incoming requests ensures efficient resource utilization, with functions executed in response to events and scaling down to zero during idle periods. This serverless approach eliminates the need for over-provisioning and guarantees optimal performance without manual intervention. Developers can focus on writing code without worrying about the underlying infrastructure.

- ***AWS Fargate***: Fargate provides container orchestration without the necessity of managing the infrastructure, but users are required to specify CPU and memory resources for each task. While Fargate offers more granular control over resource allocation compared to Lambda, it demands more manual configuration and monitoring to ensure effective resource utilization. This additional management overhead may be a trade-off for the increased control provided by Fargate.

*c. Scalability*:

- ***AWS Lambda***: The scalability of Lambda functions is automatic, responding to incoming requests and supporting virtually unlimited concurrency. Each function invocation is isolated, enabling horizontal scaling without the need to manage individual server instances. This seamless scalability ensures that applications can handle varying workloads without manual intervention, providing a highly responsive and adaptable environment.

- ***AWS Fargate***: Fargate tasks can also scale horizontally by adjusting the number of tasks in an ECS or EKS cluster. However, scaling Fargate tasks may involve more manual configuration and monitoring compared to Lambda's automatic scaling capabilities. While Fargate offers scalability options, developers may need to invest more effort in managing the scaling process effectively.

*d. Cost*:

- ***AWS Lambda***: Billing for Lambda functions is based on the number of invocations and the duration of execution, with no charges for idle time. This pay-per-use pricing model can lead to significant cost savings for applications with sporadic or unpredictable workloads, ensuring that resources are only consumed when needed. The cost-effectiveness of Lambda makes it an attractive choice for applications with varying usage patterns.

- ***AWS Fargate***: Fargate tasks are billed based on the vCPU and memory resources allocated to each task, as well as the duration of task execution. While Fargate provides more control over resource allocation, it may result in higher costs for applications with fluctuating workloads or periods of inactivity. The pricing structure of Fargate requires careful consideration to optimize costs and resource allocation effectively.

In conclusion, AWS Lambda stands out as the preferred option for event-driven, serverless architectures due to its seamless trigger integration, automatic optimization, scalability, and cost-effectiveness. However, AWS Fargate may be better suited for applications that require fine-tuned control over resource allocation or for handling long-running, persistent tasks.

IX, Quantitive evaluation

To quantitatively evaluate the performance of the proposed serverless/event-driven architectural design for the photo album application, several key metrics can be considered. Firstly, the data transfer optimization can be measured by comparing the average upload and download speeds of images and videos in the serverless architecture compared to the traditional relational database approach. Additionally, the reduction in response time can be quantified by measuring the average time taken to process and retrieve media files in the serverless architecture. This can be compared to the response time in the traditional architecture to determine the improvement achieved. Furthermore, the cost-effectiveness of the serverless architecture can be evaluated by comparing the overall infrastructure and maintenance costs incurred in both approaches. By analyzing these quantitative metrics, we can assess the extent to which the proposed serverless/event-driven architectural design enhances the performance and cost-efficiency of the photo album application.

X, Conclusion

The architecture proposed for the Photo Album application is designed to embody a sophisticated and highly efficient system that is specifically customized to fulfill the unique requirements of the company. Through the utilization of a serverless and event-driven approach on the Amazon Web Services (AWS) platform, numerous critical challenges can be effectively addressed, and the strategic objectives of the company can be closely aligned. Key components such as AWS Lambda, Amazon S3, and API Gateway each play crucial and indispensable roles in ensuring the achievement of scalability and cost-effectiveness within the system. The serverless nature of AWS Lambda enables automatic scaling, which in turn leads to the optimization of resource utilization in real-time. This capability significantly enhances the overall responsiveness of the system while simultaneously eliminating the burdens associated with manual management for the development team.

Moreover, DynamoDB, being classified as a NoSQL database, presents a rapid and cost-efficient option in comparison to traditional relational databases, which seamlessly corresponds with the overarching objective of the organization to attain data storage that is both responsive and resource-effective. The underlying framework has been intricately crafted to manage the projected twofold increase in demand occurring every six months, achieved through the effective utilization of Lambda's capabilities in managing concurrent requests and dynamically adjusting scalability, thereby ensuring optimal performance even amidst exponential expansion of the user base. Furthermore, the incorporation of Amazon CloudFront serves to elevate global response times through the strategic caching of content at edge locations worldwide, thereby contributing significantly to an enhanced application experience that is both responsive and easily accessible.

# References

AWS. (2021, 8 20). *Web Application Hosting in the*. Retrieved from https://d1.awsstatic.com/whitepapers/aws-web-hosting-best-practices.pdf.

AWS. (2024). *An AWS Cloud architecture for web hosting*. Retrieved from https://docs.aws.amazon.com/whitepapers/latest/web-application-hosting-best-practices/an-aws-cloud-architecture-for-web-hosting.html.

AWS. (2024). *Tutorial: Using an Amazon S3 trigger to invoke a Lambda function*. Retrieved from https://docs.aws.amazon.com/lambda/latest/dg/with-s3-example.html.

AWS. (n.d). *Dịch vụ điện toán - Amazon EC2*. Retrieved from https://aws.amazon.com/vi/pm/ec2/?gclid=Cj0KCQjwk6SwBhDPARIsAJ59GwdTWxaQ\_7ssPplCPC-\_xe9uFZEI0k-Zk2MepECds\_Hf--c\_KZ4Hk0EaAgkxEALw\_wcB&trk=361ccc4f-68c4-4038-bf6c-0586bee109dc&sc\_channel=ps&ef\_id=Cj0KCQjwk6SwBhDPARIsAJ59GwdTWxaQ\_7ssPplCPC-\_xe9uFZEI0k-Zk2Mep.

AWS. (n.d). *What is Amazon Rekognition?* Retrieved from https://docs.aws.amazon.com/rekognition/latest/dg/what-is.html.

C.Codex, A. (2024, 2 20). *Using Amazon S3 Event Notifications for Real-time Processing*. Retrieved from https://reintech.io/blog/amazon-s3-event-notifications-real-time-processing.

CloudDefense.AI. (n.d). *What is Identity and access management in AWS? Detailed Explanation*. Retrieved from https://www.clouddefense.ai/glossary/aws/identity-and-access-management.php.

CloudSpend. (2023, 6 1). *Mastering AWS Fargate pricing and optimizarion with CloudSpend: A comprehensive giude*. Retrieved from https://blogs.manageengine.com/cloudspend/2023/06/01/mastering-aws-fargate-pricing-and-optimization-with-cloudspend-a-comprehensive-guide.html.

K.Davis, N. (2024, 1 4). *How To Host Frontend Apps with AWS Amplify*. Retrieved from https://www.linkedin.com/pulse/how-host-frontend-apps-aws-amplify-neal-davis-ecyde/.

Kumar, P. (2023, 8 21). *Simple Notification Service (SNS) in AWS*. Retrieved from https://www.linkedin.com/pulse/simple-notification-service-sns-aws-parteek-kumar/.

Le, C. (2023, 09 25). *WHAT IS AWS? TOP BENEFITS & SERVICES OF AMAZON WEB SERVICES*. Retrieved from https://secomm.vn/what-is-aws-top-benefits-services-of-amazon-web-services/.

n.n. (n.d). *API Gateway: Express Gateway vs Amazon API Gateway*. Retrieved from https://www.express-gateway.io/eg-vs-amazon-aws-api-gateway/.

Phadke, T. (2023, 3 1). *Revolutionize Your Business with AWS Lambda and DMI: The Ultimate Serverless Solution*. Retrieved from https://www.linkedin.com/pulse/revolutionize-your-business-aws-lambda-dmi-ultimate-tushar/.

Technologies, C. (2024, 2 8). *AWS Route 53: Exploring the World of DNS and Traffic Management*. Retrieved from https://www.linkedin.com/pulse/aws-route-53-exploring-world-dns-traffic-management-cbagc/.

Varghese, P. (2023, 10 7). *Understanding and Securing Amazon Cognito: A Comprehensive Guide*. Retrieved from https://redfoxsec.com/blog/understanding-and-securing-amazon-cognito-a-comprehensive-guide/.

Werner. (2012, 1 18). *Amazon DynamoDB – a Fast and Scalable NoSQL Database Service Designed for Internet Scale Applications*. Retrieved from https://www.allthingsdistributed.com/2012/01/amazon-dynamodb.html.