Cloud Computing Architecture - Assignment 3

Serverless/Event-driven Architectural Design Report

Nguyen Gia Binh - 104219428
Student
Swinburne University of Technology
Hanoi, Vietnam
104219428@student.swin.edu.au

Bui Viet Hoang - 104060260
Student
Swinburne University of Technology
Hanoi, Vietnam
104060260@student.swin.edu.au

Pham Hung Manh - 104169507
Student
Swinburne University of Technology
Hanoi, Vietnam
104169507@student.swin.edu.au

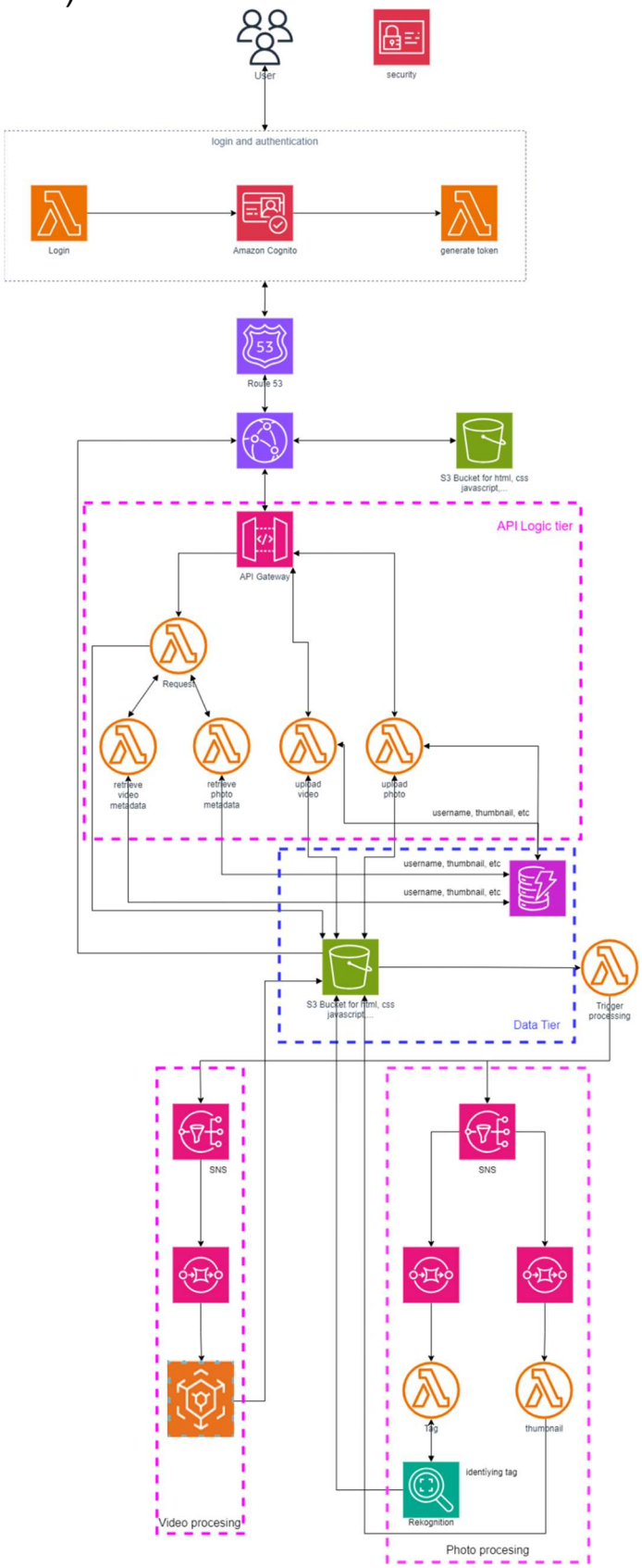
Abstract - The Photo Album application, developed previously, met with amazing success. Since then, it has been observed that the demand will grow tremendously in the future, doubling every 6 months recently. Furthermore, problems regarding speed were also mentioned, along with additional changes the company wants to make. Within this report, a serverless cloud framework is introduced, and strategically developed to cater to the demands of the Photo Album application. The design not only ensures the sustained operation of the website but also aligns with essential design criteria, encompassing performance, reliability, scalability, security, and cost-effectiveness.

I) INTRODUCTION

This document outlines a serverless cloud architecture tailored to the Photo Album application, effectively addressing various requirements. These include transitioning to managed cloud services, accommodating fluctuating demand, embracing a serverless approach, replacing the sluggish and costly relational database, optimizing global response times, and efficiently managing video and photo uploads.

AWS's service autonomy stands as a key advantage, empowering us to adopt a serverless/event-driven approach. This methodology enhances the flexibility and efficiency of our web operations. By eliminating the burden of server management, serverless computing allows us to focus on app development and implementation, freeing us from the intricacies of infrastructure administration. Additionally, event-driven architecture ensures that actions are triggered in response to specific events, fostering a more adaptable and scalable system.

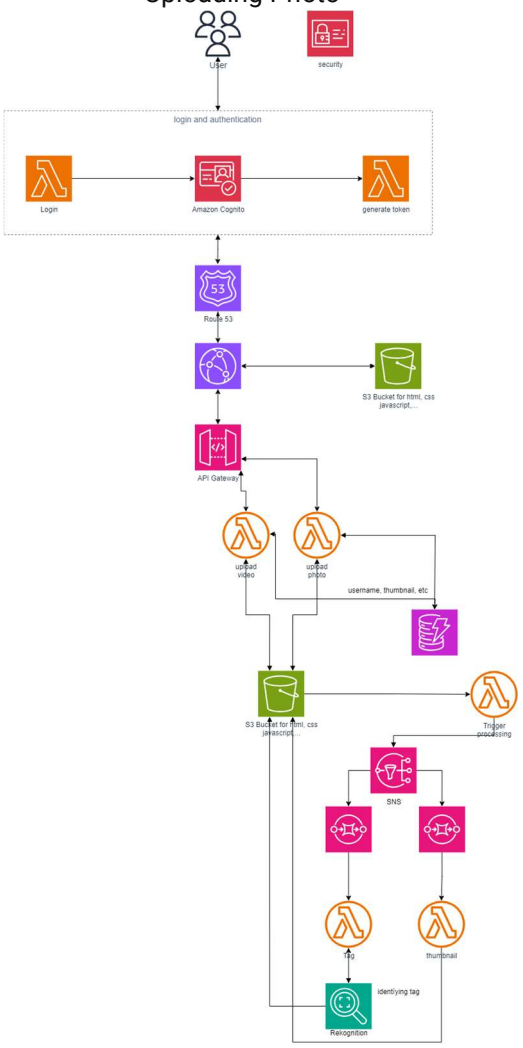
II) ARCHITECTURAL DIAGRAM



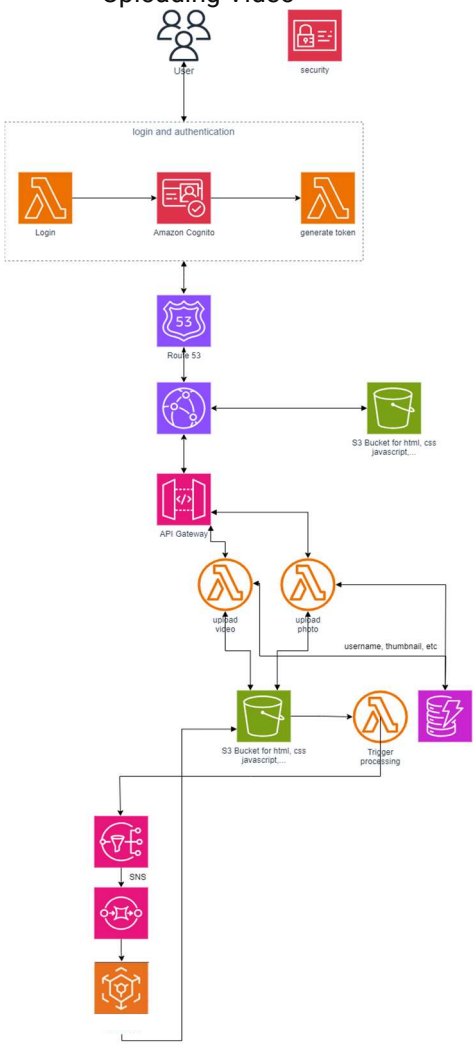
The architectural diagram shown above is the final architectural solution, as well as all AWS services used for this project.

III) UML COLLABORATION DIAGRAMS

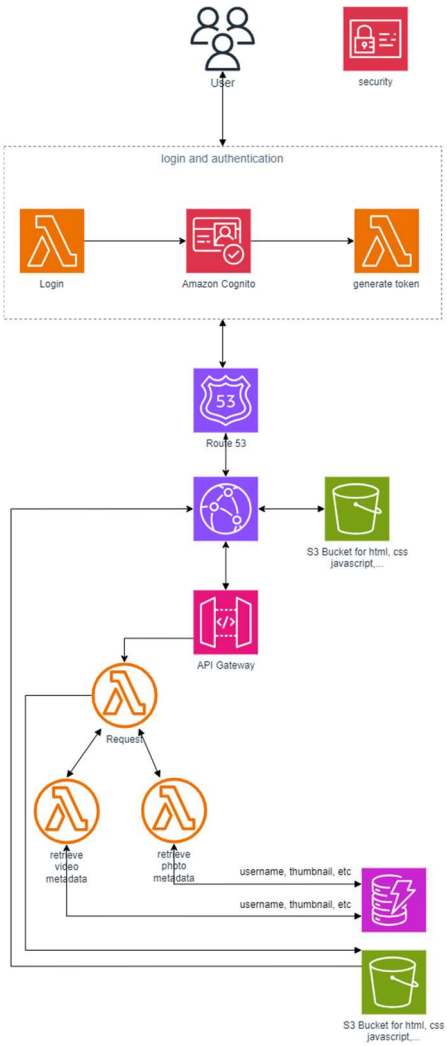
Uploading Photo



Uploading Video



Retrieving Photo/video



IV) AWS SERVICES USED

- **Route 53**
  Service Description: Amazon Route 53 is like a guide for the internet. It helps people find websites by managing the names (like www.example.com) and directing them to the right places on the internet. It's always ready to handle a lot of traffic and grow as needed.

  Purpose: In our application, we use Route 53 to give our customers an easy way to reach our service using a special web address. It's like having a custom sign for our business on the internet. Route 53 also ensures that no matter where our customers are in the world, they can always find and connect to our service without any problems.
- **CloudFront**
  Service Description: Amazon CloudFront is like a speedy courier for websites. It helps deliver web pages, pictures, videos, and other stuff to people all around the world faster. It does this by keeping copies of the content in different locations worldwide, making it quicker for people to get what they want and making more reliable content delivery.

  Purpose: In this application, CloudFront is employed to enhance response times for users globally. This is achieved by caching content in edge locations distributed across the world, reducing the time it takes for users to access and retrieve content. CloudFront also contributes an additional layer of security to the application by ensuring that content is delivered over HTTPS, enhancing the confidentiality and integrity of the transmitted data.
- **Lambda**
  Service Description: AWS Lambda is a serverless computing service that allows developers to run code without the need to set up or manage servers. Lambda functions are triggered by events, such as HTTP requests, changes to Amazon Simple Storage Service (S3) buckets, or Amazon CloudWatch events.

  Purpose: Lambda is a perfect fit for the serverless business model, allowing programmers to configure the application's internal logic without dealing with servers. It is an excellent feature for the program, as it can automatically handle additional work when a large number of users are utilizing it, ensuring that the application functions well even during peak usage. By doing this securely, it strengthens the application's security.

- **API Gateway**
  Service Description: AWS API Gateway serves as a central hub for managing APIs, simplifying the creation, sharing, and protection of APIs of any scale. It acts as the primary entry point for APIs, facilitating seamless connections between developers and various services. API Gateway also offers features to regulate API speed, manage shared resources, and ensure security.

  Purpose: Considering that the application's primary operations are handled by Lambda functions, API Gateway becomes the crucial link between the application's front-end and back-end components. For our Photo Album app, we need to accommodate a large number of visitors without compromising performance. Additionally, we prioritize reliability and security. AWS API Gateway excels in addressing these requirements by providing a scalable and reliable communication channel for the app. Moreover, it enhances security through features like controlling API request rates and restricting access to authorized users.

  Cognito
  Service description: Amazon Cognito is like a reliable assistant for developers, making it easy to add user authentication features to mobile and web apps. It can handle a large number of users and even connect with external identity providers like Facebook, Twitter, or Amazon. Cognito takes care of user management, ensuring customers can securely access their resources.

  Purpose: When combined with Lambda, Cognito acts as a gateway for users to access the application, requiring them to prove their identity. Upon user login, they receive a unique token to verify their identity within the system—an access pass granting permissions for their activities on the platform. Cognito is a fully managed service, meaning it autonomously handles and can process a large number of users.
- **Amazon Simple Notification Service (Amazon SNS)**:
  Service description: a managed service that provides message delivery from publishers to subscribers (also known as producers and consumers). Publishers communicate asynchronously with subscribers by sending messages to a topic, which is a logical access point and communication channel.

  Purpose: The application's photo and other media processing pipelines can be decoupled via the SNS service. SNS connects many application components, promoting efficient information sharing and communication. It provides feedback to different system components on detection, real-time alarms, and the state of media processing. It notifies users of newly tagged media, improving both the user's experience and the service's functionality.

- **Amazon Simple Queue Service (Amazon SQS):**
  Service description: a fully managed message queuing service that makes it easy to decouple and scale microservices, distributed systems, and serverless applications. Amazon SQS moves data between distributed application components and helps you decouple these components.

  Purpose: SNS is linked to SQS in order to decouple the application's pipelines for processing photos and videos. Jobs are queued up in SQS, and nodes can take them out to process at their own speed, avoiding overload and keeping queued jobs stored, increasing their reliability.

- **Amazon Rekognition:**
  Service description: a deep learning-powered image recognition service that detects objects, scenes, and faces; extracts text; recognizes celebrities; and identifies inappropriate content in images. It also allows you to search and compare faces.

  Purpose: Rekognition can identify tags from uploaded photos using machine learning. We don't need to create a custom model, eliminating the cost of making, training, and deploying said custom model. Plus, Rekognition is ready as is. Furthermore, it can identify faces so there are further improvements that can be made

- **Amazon Elemental Media Converter:**
  Service description: a file-based video processing service that formats and compresses offline content for delivery to televisions or connected devices. With AWS Elemental MediaConvert's high-quality video transcoding, you can create on-demand video assets for playback on virtually any device.

  Purpose: Making uploaded video to be usable across devices and platforms

- **Amazon Simple Storage Service:**
  Service description: Amazon Simple Storage Service (S3) offers secure, scalable, and highly durable object storage through virtual containers called "buckets." It allows users to effortlessly store, retrieve, and manage various data types, ensuring 99.999999999% durability through redundant storage across multiple locations within chosen regions. S3 prioritizes security with encryption, access control, and IAM integration, while its diverse storage classes cater to different needs, optimizing costs for varying usage patterns. Its versatility makes it a pivotal component for applications, enabling seamless data management, backups, content delivery, and analytics for individuals, startups, and enterprises.

Purpose: Its purpose lies in offering reliable and versatile storage for various data types, including images, videos, documents, and application data. S3 serves as a foundational element for data-driven applications, content distribution, backups, and archival purposes. It caters to a broad spectrum of use cases, from serving website content and media files to acting as a data lake for analytics and as a reliable backup solution. Its simplicity, high durability, scalability, and robust security features make it a go-to choice for individuals, startups, and enterprises seeking a robust and flexible cloud storage solution.

- **Amazon DynamoDB:**
  Service description: Amazon DynamoDB stands as a fully managed NoSQL database service within AWS, designed to provide fast, scalable, and reliable data storage for a wide spectrum of applications. Its fundamental purpose revolves around offering developers a hassle-free, high-performance database solution without the complexities of server management. It ensures consistent, low-latency access to structured and semi-structured data, making it a go-to choice for applications that demand responsive, predictable, and scalable data storage and retrieval.

  Purpose: We utilize DynamoDB to store and retrieve various attributes linked to user uploads, encompassing images, videos, usernames, and other pertinent file details. This enables us to build a responsive and scalable system that effectively organizes and handles the diverse metadata related to the content our users contribute. DynamoDB serves as the backbone of our platform, ensuring seamless management and accessibility of user-uploaded data across different aspects of our project.

V) DESIGN RATIONALE

1) Business scenarios fulfilled:

A: Minimize the need for in-house systems administration

The foundation of our architectural approach is the utilization of fully managed cloud services to lessen the burden of internal systems management. The following services have been chosen for this purpose:

- DynamoDB:

Purpose: DynamoDB has been chosen as the database solution to effectively manage and store information associated with user-uploaded images and videos.

Features: Its key-value data style and schema-less architecture make it ideally suited for this purpose, offering exceptional performance and scalability. Furthermore, the security features, multi-AZ replication, and automated scalability perfectly match the specifications.

- API Gateway:

Purpose: Serving as a secure bridge between frontend and backend functionality, the API Gateway is an essential part of the system design.

Features: It simplifies the administration, publishing, and building of APIs. Optimal performance and improved reliability are ensured through the use of automatic load balancing and fault tolerance methods. Integration with AWS Cognito and customized authentication techniques offers a highly reliable system for controlling user access.

- Amazon S3:

Purpose: Amazon S3 is the best option for hosting a static website and storing user-uploaded material.

Features: Its outstanding qualities, such as scalability, durability, and strong security features, align perfectly with the specific technical requirements. It offers scalable and long-lasting media file storage, guaranteeing the security and safety of stored material.

- Amazon Cognito:

Purpose: Amazon Cognito delivers strong identity verification and user management features to provide safe and managed application access.

Features: It efficiently manages user registrations, logins, and user data storage on a large scale. It also has the ability to support multi-factor authentication, improving overall security.

B: Rising demand

The design prioritizes scalability and dependability in order to meet the increasing demand. Serverless computing, effective storage, and content delivery are used to maximize performance.

- Lambda:

Lambda ensures seamless scalability and reliability with its automatic scaling and efficient resource utilization. It efficiently handles surges in incoming requests and is event-driven for optimal concurrency management.

- DynamoDB Auto-scaling:

DynamoDB's auto-scaling capabilities are essential for maintaining scalability. It allows the database tables to adjust read and write capacity based on real-time traffic patterns, preventing throttling issues during increased traffic.

- API Gateway Dynamic Adjustment:

The API Gateway dynamically adjusts to incoming traffic, ensuring overall scalability and reliability. With automated load balancing and fault tolerance, it optimizes performance and adapts to varying traffic patterns efficiently.

C: Serverless / Event-driven solution

AWS API Gateway serves as the entry point for all HTTP requests, directing them to the relevant Lambda functions. This not only provides a secure interface for clients but also seamlessly integrates with our event-driven model. When a user uploads or requests data, Lambda functions are triggered by events from various sources, including S3

uploads, DynamoDB changes, or custom API calls. These events are orchestrated and managed by AWS EventBridge, allowing real-time triggers and facilitating communication between different services.

In our serverless/event-driven architecture, Amazon S3 (Simple Storage Service) plays a crucial role. It serves as the foundation for storing and managing a variety of media assets essential to the Photo Application. With its scalable and durable object storage, S3 ensures high availability and low-latency access to images, videos, and other media files crucial for showcasing products. Its integration with AWS Lambda enables the triggering of functions for image processing, resizing, and other file manipulations in our dynamic content delivery pipeline. Additionally, S3's versioning and lifecycle policies automate data retention and archiving, optimizing storage costs while maintaining data integrity and compliance with our business requirements.

AWS Lambda is central to our serverless architecture, executing discrete functions in response to events within our photo application. This event-driven compute service allows us to deploy code without managing underlying infrastructure, ensuring precise scaling with demand. Lambda efficiently handles critical functionalities like photo processing, thumbnail updates, and user authentication in a serverless manner. The pay-as-you-go pricing model aligns with our cost-saving strategy, enabling us to pay only for the compute time consumed during function executions. Lambda's tight integration with other AWS services, such as API Gateway and EventBridge, facilitates smooth event handling and seamless communication across our serverless ecosystem

D: Replace the slow and costly relational database

DynamoDB's NoSQL database service stands out for its unmatched scalability and performance, making it an ideal fit for our dynamic platform's demanding needs. Its ability to handle massive workloads with consistently low latencies aligns seamlessly with our platform's requirements. By leveraging DynamoDB's flexible schema and auto-scaling capabilities, we can effortlessly manage product catalogs, user profiles, and session data while effortlessly accommodating traffic spikes during peak periods without compromising performance or incurring excessive costs.

Our decision to switch to DynamoDB stems from its inherent design, which prioritizes scalability, performance, and ease of use. Unlike our previous relational database, DynamoDB's pay-per-request pricing model eliminates the need for upfront provisioning and provides a cost-effective solution, ensuring we only pay for the resources we utilize. Additionally, its seamless integration with our serverless architecture, including AWS Lambda and Amazon S3, streamlines our workflows and enables real-time data processing, boosting overall efficiency and responsiveness.

E: Global response time needs an improvement

Route 53 for Global DNS Resolution:

- Leverage Amazon Route 53 for global Domain Name System (DNS) resolution.
- Use Route 53's global anycast network to direct users to the nearest CloudFront edge location.

CloudFront for Content Delivery:

- Implement Amazon CloudFront to cache and deliver static assets (HTML, CSS, JavaScript) and media files globally.
- Configure CloudFront to use dynamic content caching for Lambda responses.
- Utilize CloudFront behaviors and edge caching settings to optimize content delivery.

Lambda for Serverless Compute:

- Utilize AWS Lambda for serverless computing to handle dynamic content generation and business logic.

DynamoDB for NoSQL Database:

- Use Amazon DynamoDB as a NoSQL database for storing dynamic data.
- Optimize DynamoDB read and write capacity to handle the expected workload.
- Leverage DynamoDB Accelerator (DAX) for caching to improve read performance.

Consider AWS Global Accelerator as an alternative to improve global response times. It routes traffic over the AWS global network to optimal locations using static IP addresses, but it brings the following disadvantages:

- Costs: It's not free, and expenses can add up based on traffic and accelerator usage.
- Limited Control: Less control over routing compared to manual CloudFront setup.
- Learning Curve: Might be a bit complex, especially for those more familiar with traditional setups.
- Service Dependency: If AWS Global Accelerator faces issues, it can impact overall application performance.
- Features Limitation: Some features might be limited compared to using a combination of other AWS services.

F: Video Media

Since business requirements expect the system to handle video media in the future, we make a video processing system similar to the photo processing system. The video file then goes through the same procedure as the photo does before getting processed, which means the metadata of the video as well as the user_id from which the video came in will be stored in DynamoDB and the actual content of the video is stored in S3 bucket. Then a Lambda function will trigger and the video is put in the queue(SQS) waiting to be processed. We use Amazon Elemental Media Converter to transcode the video so devices can watch it across platforms, and devices with high-quality audio.

An alternative option to this is Amazon Elastic Transcoder, which provides a lower cost for a reduced monthly upload quantity and is simpler to use. On the other hand, Elemental Media Convert offers a more extensive range of features, providing control over the processing procedure, a lower price for a higher upload limit, and superior quality output.

G: Media processing

When a photo or video is uploaded to S3, the S3 bucket will send out a notification, and a Lambda function will handle the request, placing the media into its appropriate processing lane. At the beginning of each lane is an SNS node with SQS subscribed to it. For photos, there are two SQS subscriptions to one SNS, creating a parallel processing procedure. Consequently, the photo processing procedure is decoupled because each process runs in parallel and independently. Furthermore, since SNS allows for multiple subscribers, this design is scalable and extensible.

The video processing procedure follows the same architecture but only with one lane because there is only 1 requirement for it.

An alternative service to this is AWS Step Functions, It enables the coordination and visual representation of distributed application components as a sequence of state machine steps. Step Functions offers a graphical user interface for creating processes together with support for a number of AWS service interfaces, custom code, and error-handling features. But it is not included in this architecture due to its nature of complex workflows or coordinating multiple AWS services and custom logic

2) Design criteria

A: Performance, scalability (extensibility, decoupling, etc.)

- CloudFront: uses an extensive worldwide network of edge sites to cache and distribute content closer to end users, lowering latency and speeding up response times. Because it automatically adjusts to traffic variations, it can readily handle huge traffic volumes and abrupt surges in demand.

- Route 53: It is built for high availability and offers a globally dispersed network of DNS servers. It also lets you route traffic to various endpoints based on parameters like geography, latency, or set weights. By doing this, you can make sure that your DNS records are accessible and able to process large amounts of DNS queries

- API Gateway: automatically scales to accommodate large amounts of API calls. High availability is ensured by distributing traffic among several networks, and you may set up caching to lighten your backend services.

- S3 bucket: highly scalable, allowing you to store virtually unlimited amounts of data. It automatically scales to accommodate growing storage needs without any upfront configuration. S3 provides a reliable data storage solution and further guarantees great durability by storing data redundantly across many sites. S3 enables dependable and quick data transport, making it possible to upload and download big files and objects. Because of its large bandwidth capacities, it may be used to host websites, stream media, and distribute static content.

- DynamoDB: designed to scale horizontally and handle high volumes of reads and writes. It automatically partitions data across multiple servers, allowing it to scale seamlessly as application traffic increases. DynamoDB's low-latency data access makes it ideal for applications demanding rapid response times. It accomplishes this by replicating data across multiple availability zones within a region, enabling local, swift data retrieval.

- Lambda function: automatically scales your code in response to the incoming request rate. It dynamically allocates the required computing power to manage the workload and automatically scales down when demand subsides.

- Simple Notification Service: It adheres to a publish-subscribe messaging model. You can broadcast messages to a topic, and subsequently, subscribers who have subscribed in that topic receive those messages. This arrangement disentangles the message originator (publisher) from the message recipient (subscriber).

- Amazon Simple Queue Service: decouple the components of your applications. Senders and receivers do not need to directly communicate, allowing for loose coupling and improved scalability. Messages are held in queues until they are processed, ensuring reliable and asynchronous communication between components. SQS is designed for high availability and durability. It distributes messages across multiple availability zones within a region to ensure redundancy and fault tolerance.

B: Reliability

- CloudFront: Because it automatically adjusts to traffic variations, it can readily handle huge traffic volumes and abrupt surges in demand.

- Route 53: lets you route traffic to various endpoints based on parameters like geography, latency, or set weights. By doing this, you can make sure that your DNS records are accessible and able to process large amounts of DNS queries

- API Gateway: High availability is ensured by distributing traffic among several networks, and you may set up caching to lighten your backend services.

- S3 bucket: S3 provides a reliable data storage solution and further guarantees great durability by storing data redundantly across many sites. S3 enables dependable and quick data transport, making it possible to upload and download big files and objects.

- DynamoDB: DynamoDB's low-latency data access makes it ideal for applications demanding rapid response times. It accomplishes this by replicating data across multiple availability zones within a region, enabling local, swift data retrieval.

- Simple Notification Service: It adheres to a publish-subscribe messaging model. You can broadcast messages to a topic, and subsequently, subscribers who have indicated interest in that topic receive those messages.

- Amazon Simple Queue Service: Messages are held in queues until they are processed, ensuring reliable and asynchronous communication between components. It distributes messages across multiple availability zones within a region to ensure redundancy and fault tolerance.

C: Security

- CloudFront: CloudFront integrates with other AWS services, such as AWS Shield and AWS Web Application Firewall (WAF), to provide protection against DDoS attacks and other security threats.

- Route 53: supports DNSSEC (DNS Security Extensions), which adds an extra layer of security by digitally signing DNS records, preventing unauthorized modifications or DNS spoofing.

- API Gateway: provides various security features, such as authentication and authorization options, including AWS Identity and Access Management (IAM), Cognito User Pools, and custom authorizers. It also supports SSL/TLS encryption for secure communication

- S3 bucket: provides various security features to protect your data. Access to buckets and objects can be controlled through bucket policies, Access Control Lists (ACLs), and AWS Identity and Access Management (IAM) roles and permissions. Additionally, you can enable server-side encryption to encrypt data at rest.

- DynamoDB: offers robust security features to protect your data. It integrates with AWS Identity and Access Management (IAM), allowing you to control access to tables and operations at a granular level. Encryption at rest and in transit is also supported for data protection.

D: Cost

- Pay-per-use pricing: Most of the services proposed use pay-per-use pricing, so the business only pays for the resources it actually uses. As a result, this service is very cost-effective.

- No upfront fee: This service doesn't ask for an upfront fee, and it doesn't require any long-term commitment.

- Scalability: Services such as Lambda and DynamoDB automatically scale up as demand increases and scale down when demand subsides, ensuring that costs accurately reflect the resources used.

- Data Storage: S3 and DynamoDB provide efficient data storage, allowing you to save money while ensuring high availability and durability.

VI) COST CALCULATION

Service	Price Calculation(monthly)	Calculated price(monthly)
Route 53	DNS queries cost = $0.40 per million DNS queries * (1 domain name * 100 DNS queries/domain name/month) Hosted zones cost = $0.50 per hosted zone/month * (1 domain name * 10 hosted zones/domain name) Resource record sets cost = $0.05 per resource record set/month * (1 domain name * 10 hosted zones/domain name * 10 resource record sets/hosted zone)	$10.04
CloudFront	Data transfer out to origin: 1TB (free tier) Data transfer put to the internet: 1TB of data to the internet each month-> $85.00 . Requests: 100,000 requests -> 0.07$	$85.07
API gateway	Data transfer out cost Data transfer out cost = 1TB * $0.085/GB Data transfer in cost Data transfer in cost = 0 GB * $0.00/GB(Freetier) Request cost Request cost = 100,000 requests * $0.75/million requests	$85.075
DynamoDB	Data storage: 200 GB storage * $0.25/GB/month = $50.00 Read requests: 200 users/day * 5 reads/user * 30 days/month = 30,000	$52.25

	reads/month 30,000 reads/month * $0.000025/read = $0.75 Write requests: 200 users/day * 5 writes/user * 30 days/month = 30,000 writes/month 30,000 writes/month * $0.00005/write = $1.50	
S3	Storage Cost: Standard Storage Class in US East (N. Virginia): $0.023 per GB Total storage = 140 GB -> $3.22 Total data transfer = 140 GB (total storage) * $0.09/GB Data transfer cost to Southeast Asia = $12.60 Total data transfer = 140 GB (total storage) * $0.14/GB Data transfer cost to Australia = $19.60 Request cost = 24,000 requests * $0.005/1,000 requests = $0.12 Total cost = Storage cost + Data transfer cost (Southeast Asia) + Data transfer cost (Australia) + Request cost = $3.22 + $12.60 + $19.60 + $0.12 = $35.54 per month	$35.54
SNS	Free tier: 1 million mobile push notification. Request/Notification : 200 users/day * 5 request/users/day * 30 days/month = 30,000 notification * 0.00USD 1mil(free tier) / 30k (notification/month) ≈ 34 month free	$0.00
SQS	FIFO Queue: 3000 requests/month Storage Cost: 10,000 requests: $0.005 per month Monthly storage cost: (3000 requests/month) / 10,000	$0.2787

	requests * $0.005 per month Monthly data transfer out cost: 2.93 GB/month * $0.09 per GB	
Cognito	Monthly Active Users (MAUs): MAU Cost: 200 MAUs * $0.0055/MAU = $1.10 Storage Usage: 10 GB * $0.15/GB = $1.50 Custom Domain: $10 Data Transfer: For 1 TB of data transfer, the cost would be 1 TB * (1 GB/1024 MB) * $0.09/GB = $87.89.	$100.49
Lambda	Standard Lambda Architecture: x86 Total daily requests = 200 users * 10 requests = 2000 requests per day. Each request takes 300ms Memory: 512 MB Cost for 512 MB = $0.00001667 per GB-second Compute time = 900 seconds * 512 MB = 460800 MB-seconds = 0.4608 GB-seconds Cost for compute time per execution = 0.4608 GB-seconds * $0.00001667 per GB-second = $0.00000768 per execution Monthly compute time cost = 2000 executions per day * $0.00000768 per execution * 30 days = $0.4608	$0.46
Rekognition	Face Detection: $0.0012 per image * 100GB (100,000 images) Label Detection: $0.0015 per	$337.50

	image * 100GB (100,000 images) Daily Processing: Assuming processing occurs for at least an hour daily, the daily cost would be $270/24 = $11.25 Monthly Processing Cost: Assuming processing occurs daily for the entire month, the monthly cost would be $11.25/day * 30 days/month	
Elemental Media Convert	Number of users Average active users per month 200 Video processing frequency Processing events per month 30/2 = 15 Video duration Video length in seconds 60 Input video format Input video format H.264 (1080p) Output video format Output video format H.264 (720p) Audio tier Audio tier Medium Region AWS region ap-southeast-2 (Sydney) Encoding cost per minute Encoding cost for video format and audio tier in ap-southeast-2 $0.000625 Encoding cost per video Encoding cost for 1-minute video 0.000625 * 60 = $0.0375 Videos per processing event Number of videos processed in each event 2 Total processing cost per event Total encoding cost for 2 videos 2 * 0.0375 = $0.075 Monthly processing cost Total processing cost for 15 events 15 * 0.075 = $1.125	$1125
		Total: $1831.7037