COS10034 Cloud Computing
Assignment 3
Serverless/Event-driven Architectural Design Report

Table of Contents

I. ARCHITECTURAL DIAGRAM

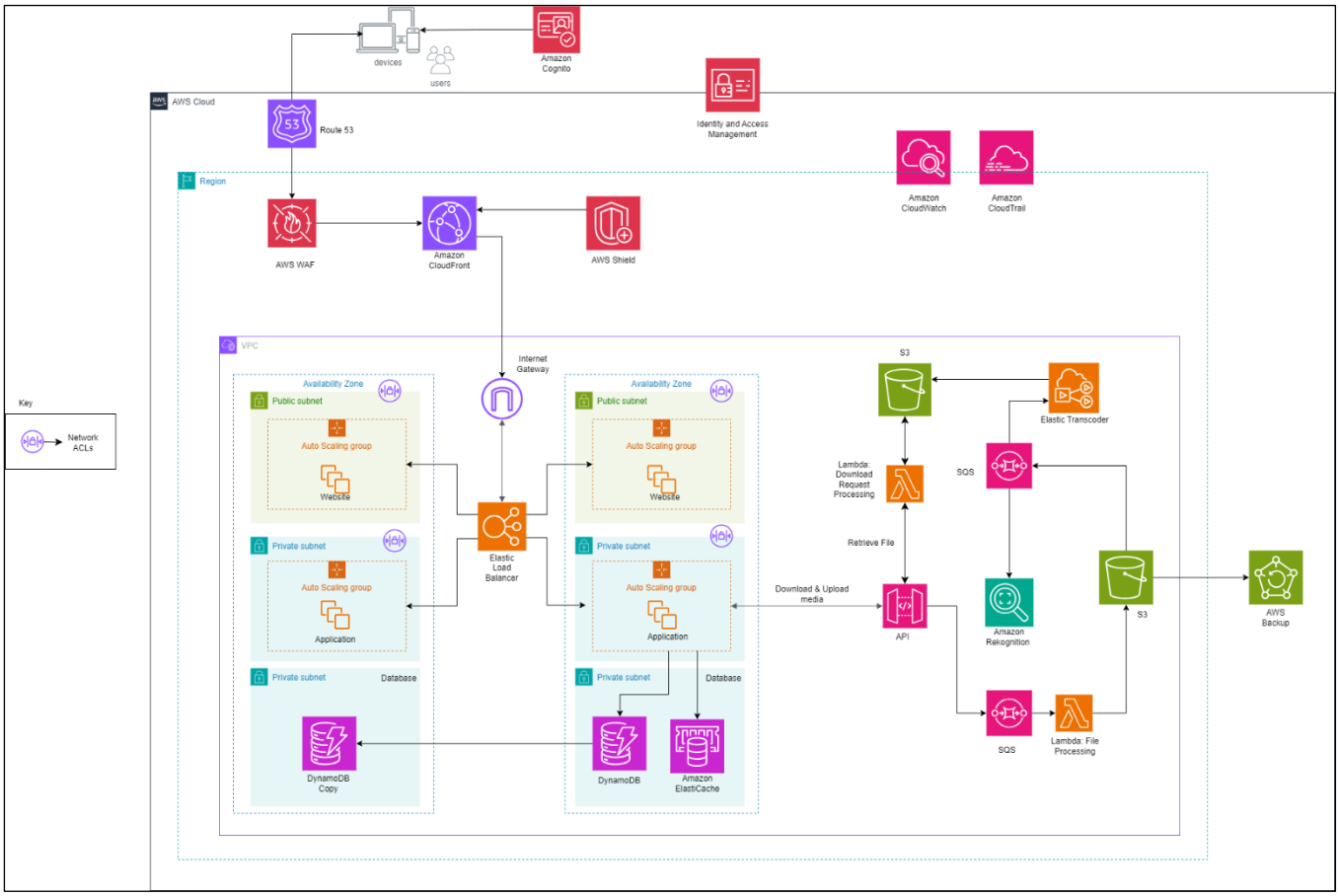


Fig. 1. Design Diagram

II. HIGH-LEVEL DESCRIPTION OF THE ARCHITECTURE

The entry point of the architecture is through the Amazon Route 53. This acts as the DNS service; it directs the user requests to the application. The AWS WAF and the AWS Shield provide a layer of protection against harmful web threats and DDoS attacks. Amazon Cognito manages user authentication and access. This makes sure the sign-up and sign-in processes are done correctly and securely.

Amazon CloudFront serves as a CDN (Content Delivery Network). It caches content at edge locations for low-latency delivery to global users.

We have used Amazon Virtual Private Cloud (VPC), to encapsulate the architecture and provide an isolated area in the AWS Cloud with configured subnets, route tables and gateways. The Internet gateway connects the VPC to the internet. It allows communication between instances in the VPC and the outside world. We have also used an Elastic Load Balancer that

distributes the incoming traffic load across multiple targets in the Auto Scaling groups to balance the load and improve availability.

Auto Scaling Groups are used to adjust the number of Amazon EC2 instances in response to the application's load. This makes sure that resources are used efficiently, and performance is upheld.

We use the database services, Amazon DynamoDB. This service provides a fast and flexible NoSQL database for storing and retrieving data. We have also used Amazon ElastiCache. This is used to cache the data that is accessed the most frequently. This will help to reduce the load on the database and also improve response times.

AWS Lambda Executes code in response to triggers (like file uploads to S3). This will be handling various backend processes without having to allocate servers. The Amazon S3 Bucket stores and retrieves any amount of data. This serves as the primary data storage service for the application's media files. We have used AWS Elastic Transcoder, which converts media files into different formats suitable for various devices and platforms. Then Amazon Rekognition provides image and video analysis for features like tagging and content moderation. It also allows for future implementations of AI used to automatically identify tags in photos.

Amazon SQS decouples components by using message queues. This will make sure that the system can handle bursts of traffic or workload increases without losing messages, tasks, or overwhelming processing services. AWS Backup gives a centralized backup service that makes sure data is recovered in the case of a data loss.

Furthermore, with the use of Amazon CloudWatch we can monitor the performance and health of the application, providing logs and metrics. We use Amazon CloudTrail to record AWS API calls for each account. This allows for governance, compliance, and operational and risk auditing.

Finally, we have made use of AWS Identity and Access Management (IAM), which is used to control access to all the services. IAM will be integrated along with Amazon Cognito for giving out user authentication and access to services. IAM plays a critical role in managing who can do what across all AWS services being used.

III. Description and Justification of the Services

A. Amazon Virtual Private Cloud (VPC):

This service allows us to isolate resources into a private network space. It allows for our infrastructure to exist in.

Justification:

This service provides a safe and secluded place for our resources. We can manage our network and its security with this service. It allows us to create subnets, configure route tables and network gateways. This also enables the enforcement of security best practices using Network Access Control Lists (NACLs) and allows us to edit security group settings which is very important for any infrastructure.

B. AWS Auto Scaling:

This service helps us automatically adjust the capacity of our networks to keep a consistent performance at a very low cost.

Justification:

Due to the huge growth of the company, there is a need to handle increasing load. Because of this Auto Scaling comes into good use since it will make sure that resources are matched to the demand without manual intervention. The cost is managed at a low level as well since it can reduce capacity when there is less demand.

C. Amazon Route 53:

This service is a scalable DNS web service which is supposed to give a reliable way to route user requests to the applications.

Justification:

This will give us DNS services for our domain. It also gives us the ability to route user traffic to the nearest endpoint via global DNS resolution. This will reduce latency and handle global traffic effectively.

D. Elastic Load Balancing:

This will automatically distribute the incoming application traffic evenly across multiple targets (EC2 instances).

Justification:

This is used to ensure that the system performance is maintained and prevents one instance from being overloaded, since it can scale incoming traffic. This makes sure that there's fault tolerance in our applications; it can detect unhealthy instances and automatically reroute traffic to the healthy instances. This will reduce the risk of downtime. Since the company is experiencing growth, these functionalities are useful.

E. AWS Identity and Access Management (IAM):

This IAM service allows us to manage access to AWS services and resources securely by defining user policies with permissions. IAM roles with specific permissions would be attached to AWS resources, like Lambda functions, allowing them to access other AWS services (For Example: writing logs to CloudWatch, reading and writing to S3 buckets) securely.

Justification:

This is important because we need to secure our services and data. This service will let us control who can access what resources which gets rid of unauthorized access therefore removing potential breaches in security.

F. Amazon CloudFront:

This is a CDN service (fast content delivery network). This lets us deliver our media and content to customers globally as it uses edge locations which are strategically placed across the world. When a user requests content, they will be directed to the nearest edge location. This will reduce latency since the distance between the user and the data center is reduced.

Justification:

By ensuring that users in any location in the globe can access content with the same efficiency, this service is particularly beneficial since this company is experiencing global growth and have international customers (For example, like users in Australia).

G. Amazon CloudWatch:

This service is responsible for monitoring the health and performance of the AWS environment. Using this service we can see the use of resources, the performance of the applications and the health of operations. It collects data like CPU usage and data transfer rates which is critical to make sure that the resources are used optimally.

Justification:

Since this infrastructure needs to scale rapidly to meet a doubling demand every six months, the ability to monitor, collect and analyze metrics becomes very important for operations to run in an optimized manner.

H. Amazon CloudTrail:

This service gives a detailed report of every record of API calls made within an AWS account. Detailed logging helps to reduce potential security threats by continuously monitoring, analyzing, and logging account activity across the architecture.

Justification:

As the company increases in volume, the number of users interacting with AWS resources will increase as well. Because of this the risk of misconfigurations and the risk of unauthorized access will increase too. So, this service will prove to be invaluable in the face of these increased potential security threats.

I. AWS Lambda:

Lambda lets the architecture be completely event-driven. It can be triggered by AWS services; for this instance, an upload to the Amazon S3 bucket. This kind of event can trigger a Lambda function to occur. It can also scale automatically to handle the number of requests. This service is used to initiate a series of tasks when media is uploaded, such as transcoding video files with AWS Elastic Transcoder.

Justification:

This serverless method can reduce the operational complexity (being event driven) and is also very cost effective. It is cost effective since we are charged per the number of requests for the functions and the time taken for the function to execute.

J. Amazon Simple Storage Service (S3):

The S3 bucket service gives secure object storage which enables us to store and retrieve any amount of data. It is basically a general-purpose storage service with high durability and high availability.

Justification:

Since we need to adapt to the company's rapid growth and high demand, the scalability of the S3 service is essential to us. The high availability and durability of the S3 buckets also factor in to how vital this service will be as we need to store a lot of media uploaded by the user.

K. Amazon Cognito:

This service provides authentication and authorization of users along with user management for applications. Users can sign in directly or through third parties. Amazon Cognito creates user pools that manage sign-up, sign-ins and grants users access to other AWS services.

Justification:

We need this service for user identity management since our applications have a big user base. It allows user registration, authentication and account recovery which is important when it comes to managing potentially many numbers of global users. Overall, when it comes to our mobile and web applications, this service is useful to provide a secure user experience.

L. Amazon DynamoDB:

Amazon DynamoDB is a NoSQL database that is fully managed and durable with security built in. It is used for storing structured data such as user profiles, file metadata, activity logs, subscription data, application state and other operational data. It's scalable and quick and the data is highly accessible.

Justification:

Since this service offers low-latency performance at any scale, this is important since we have applications that need fast access to data like user profiles, application state etc. We can use an on-demand paying model, which can be cost-effective if we monitor it closely. Since there's no management of hardware or software needed, compared to a traditional RDBMS, DynamoDB can be cost-effective.

M. AWS Elastic Transcoder:

This is a media transcoding service. This allows us to convert media files, transcode video files, convert media into low resolution versions for mobile phones, and create thumbnails.

Justification:

Since the company requires creating thumbnails and changing the resolutions of video files, this service is essential to fulfill that criterion. It is a pay-as-you-go price model, so we only pay for the actual usage, which is important in maintaining cost efficiency with changing demand.

N. AWS Rekognition:

AWS Rekognition makes it easy to add image and video analysis to our applications. It can identify objects, people, text, scenes, and activities in images and videos, as well as detect any inappropriate content. Rekognition also provides highly accurate facial analysis and facial search capabilities to detect, analyze, and compare faces.

Justification:

As the architecture needs to be extensible for additional AI solutions for identifying tags in photos, AWS Rekognition is perfect for this criterion. This is particularly useful for automatically enhancing media with metadata, improving searchability, and creating a more engaging user experience.

O. Amazon Simple Queue Service (SQS):

This service is a message queuing service. It can be used to decouple sending and receiving components, without requiring each component to be concurrently available. Basically, when one service wants to send a task to another service, it isn't given directly, instead it is loaded into the queue in the SQS. This service is important since 2 parts of the applications don't have to be free at the same time. If the part of the application to which the task is meant to be sent to is busy, the task will remain in queue until the part becomes free. This way tasks are not lost if one part of the architecture is busy.

Justification:

SQS is critical for managing communication between decoupled components in a cloud architecture. It acts as a message buffer, ensuring that request handling components, such as AWS Lambda functions, do not lose messages when there are sudden spikes in demand. SQS queues can store messages until the next service is ready to process them, which is important for maintaining the responsiveness of the application during peak loads. This decoupling also improves the fault tolerance of the system; if one component fails, the messages remain in the queue, preserving data integrity.

IV. How the business scenario is fulfilled using the proposed services

Firstly, we have made use of AWS S3 services for media storage. As this is a managed service, we have decreased the need for in-house management of storage systems.

We have made use of Auto Scaling groups for EC2 instances and as AWS Lambda is serverless as well, we fulfill the requirement of creating a system that can automatically scale according to increasing demand and growth.

To manage the performance of EC2 instances and ensure computing capacity aligns with demand, we implement AWS Auto Scaling. This monitors the application and auto adjusts the number of EC2 instances to respond to the changes in demand. This makes sure that the number of instances in service increases when the load approaches the upper limit of 60% and decreases when the load reduces. By resizing the instances to an instance type of t3, we can improve performances greatly. This will also help to maintain the traffic load within the targeted range of 50-60%.

Using AWS Lambda to provide an event-driven solution that triggers processes in response to certain events, such as uploading media to S3, we can make the architecture a more serverless model. This eliminates the need to monitor and adjust EC2 capacity manually, as Lambda functions are designed to handle bursts of traffic and run only when needed, which can lead to cost savings and operational simplicity.

We address the need to improve global response times by implementing Amazon CloudFront as a CDN. It is designed to deliver content from the edge locations closest to users.

With the use of AWS Elastic Transcoder, we can handle tasks such as video transcoding. It can convert media files into different versions optimized for various devices, such as smartphones, tablets, and computers. Essentially, this lays the foundation for the company to go into video media, without much more investment.

The architecture that we have designed allows the automatic production of various media versions using AWS Lambda and AWS Elastic Transcoder. This process is initiated when there is a media upload to S3. This media upload triggers an event notification, which is detected by Lambda. Lambda then invoked a function which starts the media processing task. Lambda calls AWS Elastic Transcoder to begin converting the media files into different versions (like lower resolution for mobile devices and creating thumbnails for previews).

This system is also extensible to the use of future AI tagging features since we have implemented AWS Rekognition. After a media file is transcoded, another Lambda function can trigger Rekognition to analyze the content and extract metadata, such as identifying objects, scenes, or text within images and videos.

Since we use serverless services Lambda and SQS and Auto Scaling Groups, the system will naturally scale according to the incoming load. SQS will make sure that the system is not overloaded by queueing tasks. SQS effectively decouples the architecture, allowing services to process jobs from the queue without overloading the application. This ensures that no requests/tasks are lost due to an increase in uploads. If there are highly demanding tasks, we can make use of the scalable EC2 instances.

How decoupling of the architecture is achieved:

The architecture achieves decoupling through a series of AWS services that interact via event-driven triggers and message queues, which allows for components to operate independently, and scale as needed.

AWS Lambda functions respond to events, such as file uploads, and process data. Amazon SQS acts as a buffer, holding messages that dictate tasks for the next services. Amazon S3 provides a central storage solution that other services can access on-demand, without direct dependencies between services.

Elastic Load Balancing distributes traffic across EC2 instances in Auto Scaling groups. This will be making sure no single instance is critical to the application's uptime. Amazon DynamoDB and ElastiCache offer high-performance data access that services can use independently. AWS Elastic Transcoder queues and processes media files automatically, initiated by Lambda.

Lastly, AWS Backup conducts backup operations discreetly, ensuring data durability without impacting active data processing workflows.

This decoupled configuration makes maintenance easier, increases system durability, and allows for easier scalability.

V. UML Collaboration Diagrams

A. Uploading

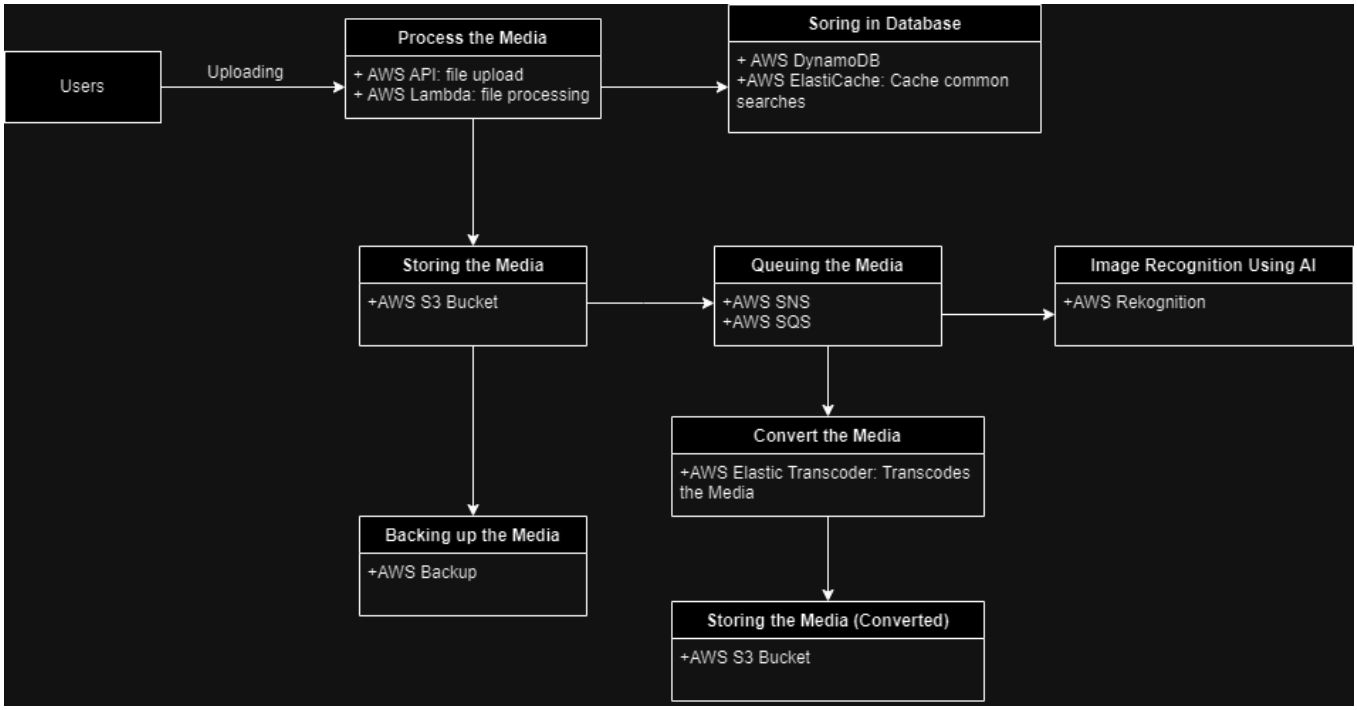


Fig. 2. UML Diagram of Uploading

B. *Downloading*

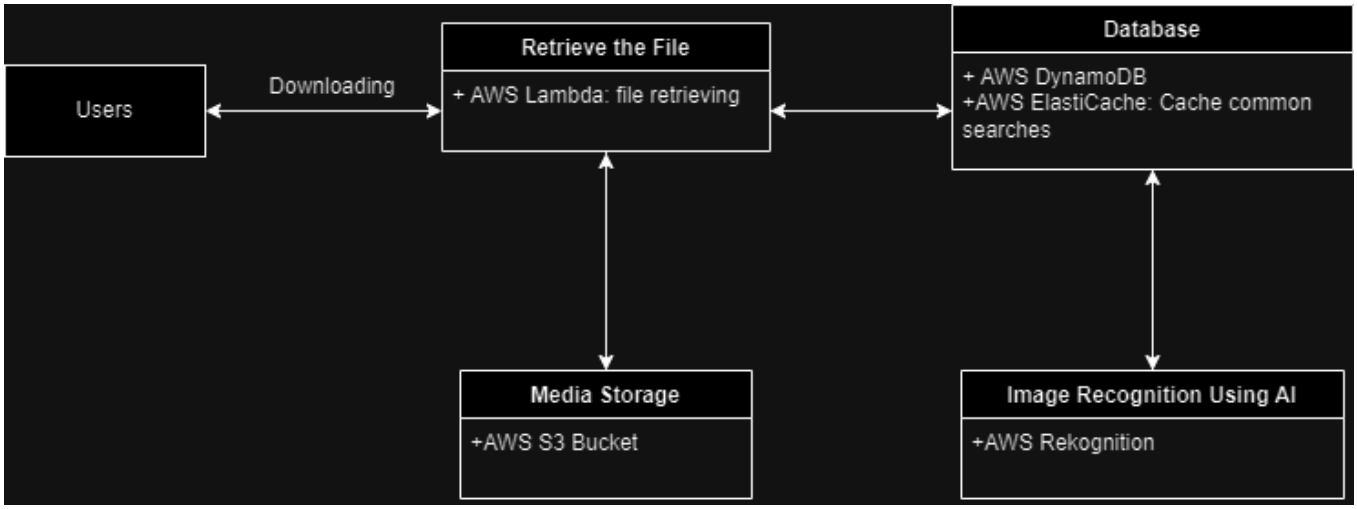


Fig. 3. UML Diagram of Downloading

VI. UML Sequence Diagrams

A. *Uploading*

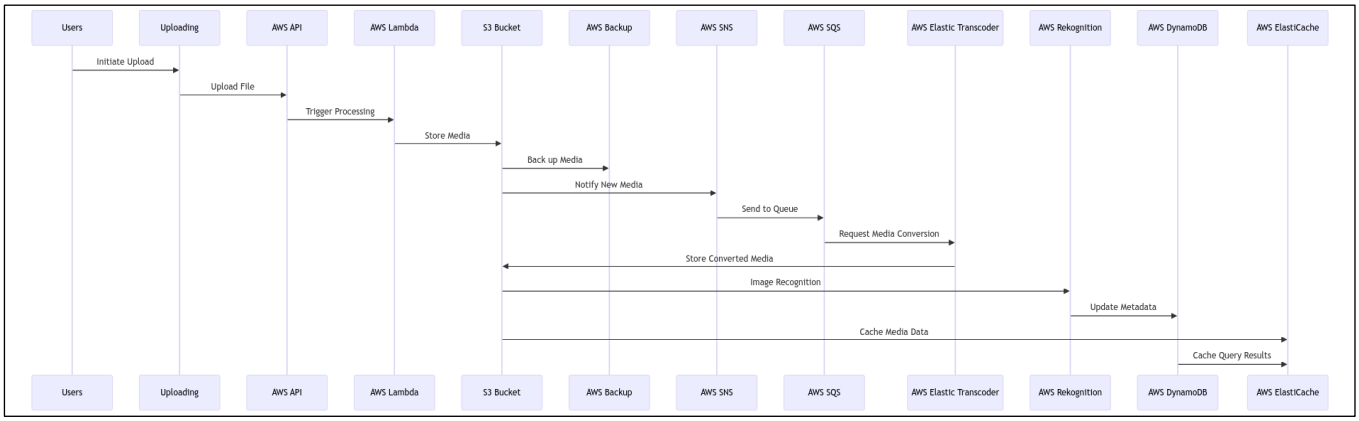


Fig. 4. UML Timeline of Uploading

B. *Downloading*

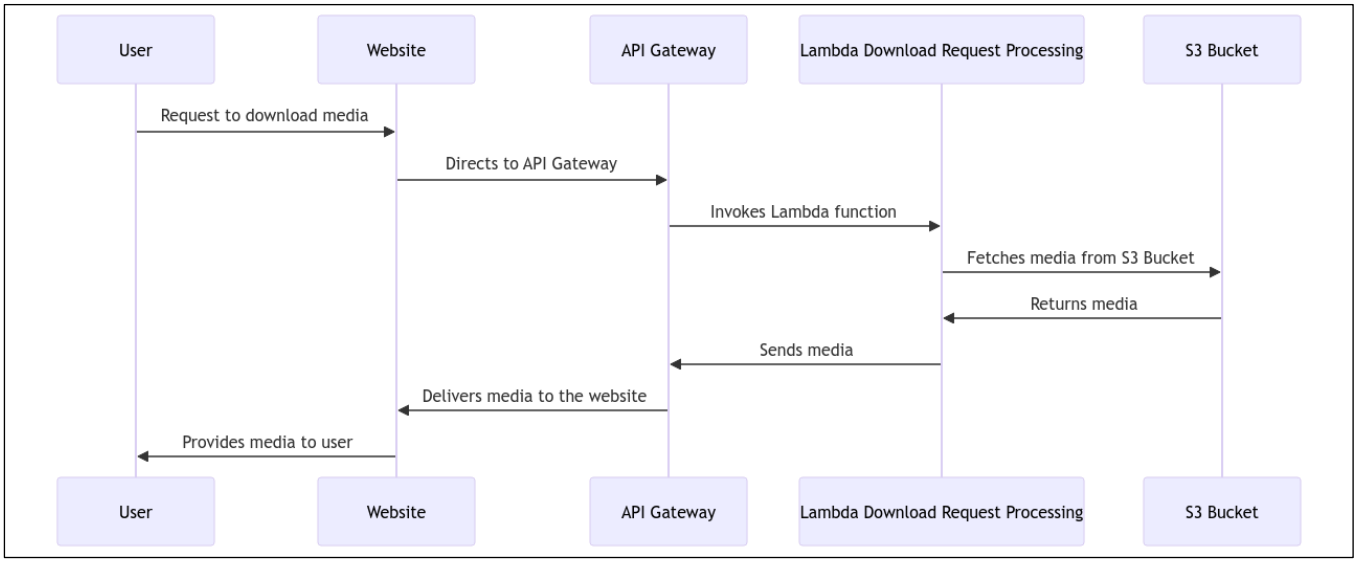


Fig. 5. UML Timeline of Downloading

VII. Design Rationale

A. Alternative Solutions and Comparisons

1) Virtual machines vs. Containers vs. Serverless computing

Virtual Machines: EC2 instances function as dedicated servers and offer control over the computing environment, which can lead to higher costs if not managed efficiently. They do not automatically scale, which require manual intervention, making them suitable for applications with specific legacy or compliance requirements.

Containers: Using Amazon ECS for container management allows for streamlined deployment and management of applications. However, it may not automatically adapt to sudden increases in traffic as effectively as auto-scaled environments, which could require manual intervention.

Serverless Computing: AWS Lambda operates on a serverless model, providing a scalable and event-driven environment with a cost structure that aligns with usage. This model is optimal for workloads that are event-triggered and do not require much computing power.

2) SQL vs. NoSQL database

SQL Databases: Amazon RDS offers a traditional relational database model with powerful querying capabilities but may need effort to scale horizontally.

NoSQL Databases: DynamoDB offers flexibility and is designed for rapid scalability and high performance, ideal for less complex queries and high throughput needs.

3) Caching Options:

In-memory Caches: Services like ElastiCache offer fast data access and are suitable for real-time analytics and session storage.

CDNs: CloudFront accelerates content delivery by caching content closer to users, ideal for static and dynamic content delivery across the globe.

Database Caching: Caching within the database layer can reduce query load but may offer less flexibility and slower performance than in-memory solutions.

4) Push vs. Pull message handling options

Push-Based Messaging: Services like SNS can push messages to consumers in real time but risk overloading them if not carefully managed.

Pull-Based Messaging: SQS allows consumers to process messages at their own pace, providing a more controlled environment and preventing overloading.

5) Number of tiers in the architecture

3-tier Architecture: This architecture is structured in three layers — one for user interface, one for the business processes, and one for data management. Each layer runs independently, which can improve security and allow each part to grow or shrink based on demand. While this setup is more complex to manage, it provides the advantage of fine-tuning each layer separately for better control and protection.

2-tier Architecture: In a 2-tier setup, the user interface and business logic layers, or the business logic and data layers, are combined. This can lead to faster communication between these layers since there are fewer divisions within the architecture. However, for more complex systems, this can lead to performance issues as the combined layers may struggle to handle heavy loads efficiently.

B. Design Criteria:

How our architecture fits the design criteria:

Performance and Scalability: The use of Lambda and Auto Scaling ensures the system scales with demand while maintaining performance. DynamoDB and ElastiCache address high throughput and low latency needs.

Reliability: Decoupling with SQS, redundancy with S3, and multi-AZ deployment of EC2 instances contribute to high system reliability.

Security: IAM provides comprehensive access control, while AWS WAF and Shield protect against threats (such as DDoS attacks). Cognito secures user authentication.

Cost: Serverless computing, and NoSQL databases can be more cost-effective due to their pay-as-you-go pricing and managed nature.

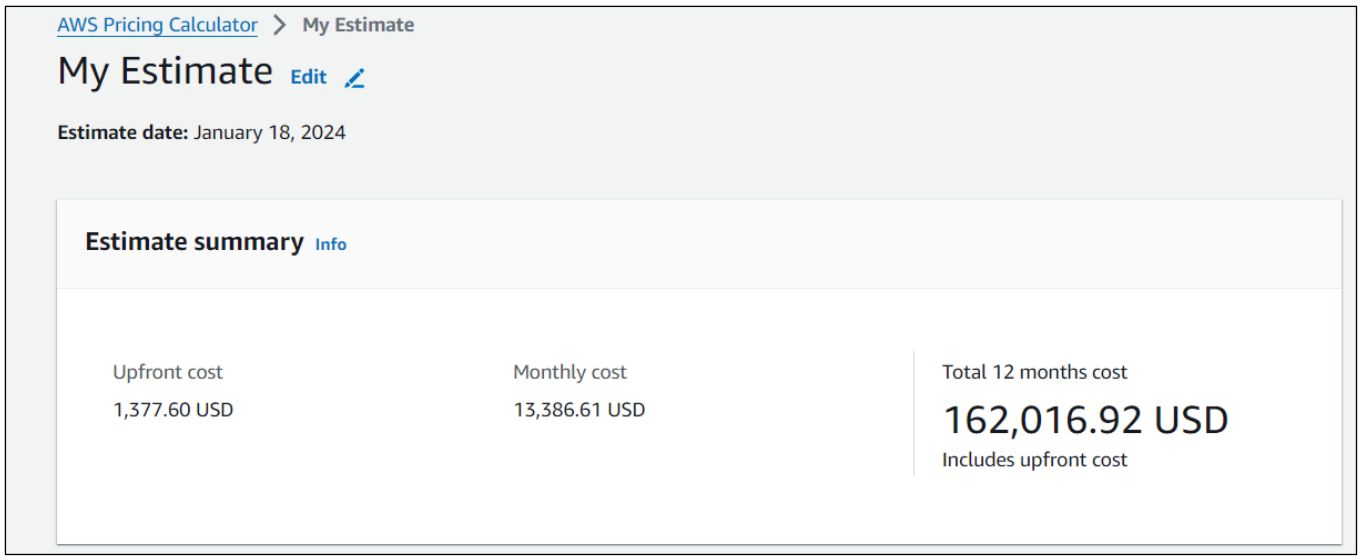
C. Cost Summary

Using these AWS services, we have strived to form an architecture that's as cost-efficient as possible. We have calculated a rough estimate of how much it will take monthly, yearly and the upfront cost that will be needed. We used the service AWS Pricing Calculator to determine this rough estimate of the costs.

Around $1,377.60 will be required for the initial upfront cost to set up all the services. After this, the estimated monthly recurring cost for our architecture will be $ 13,386.61. The total estimate cost for the year, including the upfront cost, will amount to $ 162,016.92.

Below is a summary of the specific costs for the services that we have used:

- Amazon Cognito: $514.25 per month for user authentication and management services.
- Amazon Route 53: $50.00 per month for DNS services.
- Amazon CloudWatch: $30.00 per month for monitoring services.
- AWS WAF: $148.60 per month for web application firewall services.
- Amazon CloudFront: $609.52 per month for content delivery network services.
- AWS Shield: $3,000.00 per month for additional protection against DDoS attacks.
- Amazon DynamoDB: $501.27 per month for the NoSQL database services.
- Amazon ElasticCache: $101.47 per month for in-memory caching services.
- Amazon S3: $1,177.60 per month for object storage services.
- AWS Elastic Transcoder: $4,500.00 per month for media transcoding services.
- AWS Lambda: No direct costs, it is included in the request or data transfer fees.
- Amazon API Gateway: It is a pay-per-use pricing model based on API calls.
- Amazon SQS: No direct costs as it depends on other service usage.
- Elastic Load Balancing: $33.00 per month for distributing application traffic.
- Amazon Rekognition: $1,600.00 per month for image and video analysis services.
- Amazon VPC: $66.34 per month for network infrastructure.
- Amazon EC2 Instance: $241.19 per month.



VIII. Task Allocation

Hasindu – Designing the Architecture Diagram and explaining the services used and explaining the architectural diagram.

Radin – Comparing alternative solutions and explaining how our architecture fits the design criteria.

Thirath – Creating UML Collaboration and Sequence Diagrams for the design and explaining how the business scenario is fulfilled using the proposed services.

IX. Conclusion

In conclusion, our architecture successfully meets our design criteria, delivering on performance and scalability through AWS Lambda and Auto Scaling, which enable our system to adapt to user demand dynamically. Amazon DynamoDB and ElastiCache enhance data handling, ensuring quick access and high throughput. Reliability is ensured via the strategic use of SQS for task management, S3 for durable storage, and EC2 across multiple availability zones. Security is rigorously managed with IAM, AWS WAF, Shield, and Cognito safeguarding our system from threats and unauthorized access. Additionally, the cost-efficient model of serverless and NoSQL technologies provides a financially sustainable solution. Overall, our design establishes a solid and scalable foundation, ready to support the evolving needs of our cloud-based applications.

X. Acknowledgment

We are deeply thankful to the academic staff of Swinburne University of Technology, particularly mister Tharindu Suraj, whose invaluable guidance and insights have been instrumental in shaping our architectural design. We express our appreciation to the university for providing the necessary resources and tools that have facilitated our research and design process. We extend our sincere gratitude to Amazon Web Services (AWS) for providing a robust and comprehensive suite of cloud services that have been important in the development and deployment of our infrastructure.

XI. References

[1] “AWS Pricing Calculator.” [Online]. Available: https://calculator.aws/#/addService

[2] “Are You Well-Architected?,” Amazon Web Services, Inc. [Online]. Available: https://aws.amazon.com/architecture/

[3] “AWS-CloudDesignPattern.” [Online]. Available: https://en.clouddesignpattern.org/index.php/Main_Page

[4] Amazon Web Services. (Year, Month). "AWS Web Hosting Best Practices" [Online]. Available: https://d1.awsstatic.com/whitepapers/aws-web-hosting-best-practices.pdf

[5] Amazon Web Services. (Year, Month). "Well-Architected Framework" [Online]. Available: https://docs.aws.amazon.com/pdfs/wellarchitected/latest/framework/wellarchitected-framework.pdf

[6] "NoSQL vs. SQL Databases," MongoDB. [Online]. Available: https://www.mongodb.com/nosql-explained/nosql-vs-sql

[7] "Understanding the Different Types of Load Balancers," Solve The Network. [Online]. Available: https://blog.solvethenetwork.com/different-type-of-load-balancers/

[8] “An AWS Cloud architecture for web hosting - Web Application Hosting in the AWS Cloud.” [Online]. Available: https://docs.aws.amazon.com/whitepapers/latest/web-application-hosting-best-practices/an-aws-cloud-architecture-for-web-hosting.html

[9] “Key components of an AWS web hosting architecture - Web Application Hosting in the AWS Cloud.” [Online]. Available: https://docs.aws.amazon.com/whitepapers/latest/web-application-hosting-best-practices/key-components-of-an-aws-web-hosting-architecture.html

[10] “Key considerations when using AWS for web hosting - Web Application Hosting in the AWS Cloud.” [Online]. Available: https://docs.aws.amazon.com/whitepapers/latest/web-application-hosting-best-practices/key-considerations-when-using-aws-for-web-hosting.html