COS20019 – Cloud Computing Architecture
Assignment 3
Serverless/Event-driven Architecture Design Report

Nur E Siam-103842784
Bachelor Of Computer science
Swinburne University Of Technology

Ruffin Remad-103840173
Bachelor Of Computer science
Swinburne University Of Technology

Promit Prosun Barua-104183675
Bachelor Of Computer science
Swinburne University Of Technology

Abstract
Cloud computing has become increasingly popular in the development of online architectures. Services like Amazon Web Services (AWS) enable developers to utilize managed services that take care of the underlying infrastructure, allowing them to concentrate on application logic rather than dealing with servers, networking equipment, and system configuration. In this paper, we will present a proposal for designing and implementing a serverless and event-driven architecture to enhance the performance and efficiency of a business based on their specific scenario. This architectural proposal will make use of AWS services such as Lambda, API Gateway, DynamoDB, SQS, and others. In addition to outlining the architecture design, the paper will also delve into the reasoning behind the design choices and provide justification. Furthermore, the paper will explore some noteworthy alternatives to the proposed architecture.

Keywords: Cloud computing, AWS, Serverless/ Event-driven, business scenario, design rationale.

1.Introduction

Cloud computing has become a revolutionary force for enterprises in the modern digital environment. The market for cloud applications is expected to reach a value of $153.6 billion by 2023, demonstrating its unparalleled significance. From $30.4 billion in 2013 to above $100 billion in 2018, the market has grown remarkably year over year. According to projections, the market for cloud applications would grow to a remarkable $168.6 billion in value by 2025. In order to solve critical issues including infrastructure management, scalability, database optimisation, cross-region performance, and media processing, this paper goes into an architectural concept for a cloud-based picture album application. This approach makes use of a variety of AWS services.

2.Business Scenario

The Photo Album application has emerged as a ground-breaking platform in the age of digital transformation and has had extraordinary success, prompting ongoing development to satisfy the rising demand. It need ongoing improvements to meet the needs of a user base that is expanding. This paper describes identified needs and problems and suggests solutions to improve the customer experience. These include media processing evolution and optimisation, scalability, database optimisation, and infrastructure management. The resulting design satisfies these demands and includes further discussion to test its efficacy and performance with the ultimate goal of giving users an unmatched experience:

2.1 Infrastructure management:

- Utilize managed cloud services to reduce in-house system administration and storage overhead.

2.2 Scalability:

- Design an architecture capable of accommodating expected growth.
- Adopt a serverless/event-driven solution for enhanced scalability.

2.3 Database Optimization:

- Explore cost-effective options for the relational database while ensuring performance.

2.4 Cross-Region Performance:

- Enhance global response times to ensure an exceptional user experience.

2.5 Media Processing Evolution and Optimization:

- Prepare for video media handling in the future.
- Implement automated media version generation upon upload.
- Ensure an extensible architecture for media processing.
- Effectively decouple the architecture to prevent overloading.

3. AWS Architecture Diagram

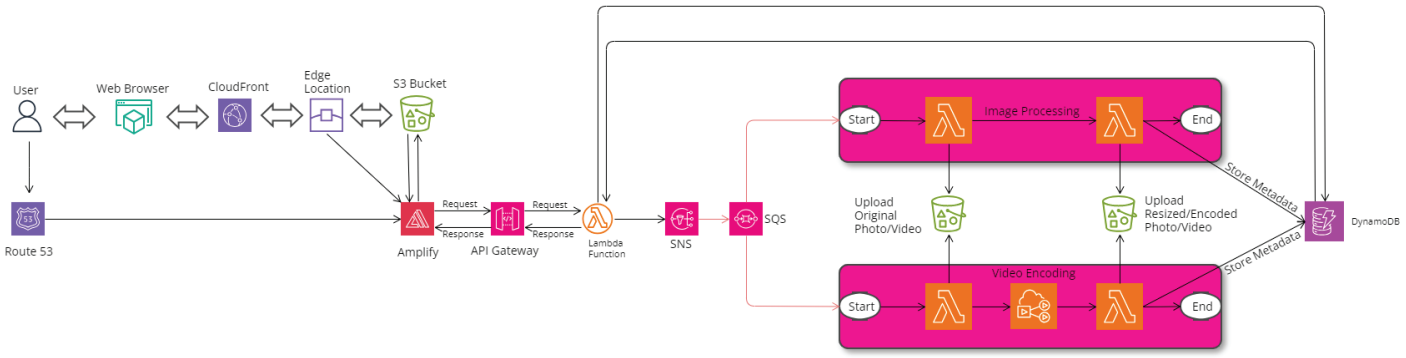

Figure 1: Architecture Diagram

3.1 AWS Content Delivery System

The AWS Content Delivery System leverages AWS Route 53 and AWS CloudFront to enhance the application's global reach and content delivery experience. AWS Route 53, a scalable Domain Name System (DNS) web service, facilitates domain registration and efficiently routes user requests to AWS resources across the world. AWS CloudFront, a content delivery network (CDN) service, accelerates content delivery by caching data at edge locations, significantly reducing latency and ensuring a seamless user experience.

3.2 AWS Serverless Three-Tier Architecture

The following list outlines the functions of the major services inside the AWS Serverless Three-Tier Architecture:

Amazon S3: Amazon S3 acts as the storage layer, safely storing user data and media assets. It guarantees great availability and sturdiness, which makes it the perfect option for data storage.

AWS Amplify: With capabilities like authentication, API integration, and more, AWS Amplify makes frontend development simpler. The development process is streamlined, saving time and resources.

API Gateway: By serving as the entry point for client queries, API Gateway makes it easier for clients to communicate with backend services. With capabilities like authentication and throttling, it controls API requests and routes them to the proper services.

AWS DynamoDB: The NoSQL database used by DynamoDB to store application data. It is appropriate for real-time applications because it offers low latency access, automated scalability, and seamless connection with other AWS services.

The serverless architecture has several benefits, one of which is auto-scalability. Cost-efficiency results from ensuring that resources are automatically modified in accordance with demand. Additionally, it drastically lowers operational costs, freeing developers to concentrate on code and innovation rather than infrastructure maintenance.

3.3 Reliable Fanout Architecture

The effective data distribution provided by the fanout architecture ensures that postings or changes are swiftly sent to all pertinent users or clients in real-time. This approach is very helpful for social networking platforms and other applications where user posts must rapidly reach their following. To send out notifications whenever a new post or change is made, AWS SNS is used. These alerts are briefly stored in AWS SQS's message queue, which distributes them to several AWS Lambda processes at once. These Lambda functions are in charge of updating the user's feeds in accordance with the alerts they have received, providing prompt and dependable information delivery.

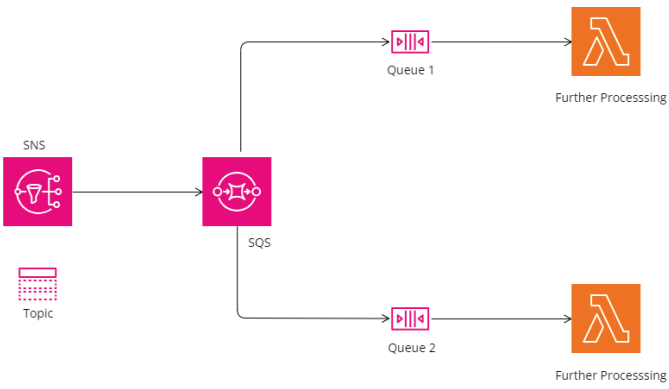

Figure 2: Fanout Architecture

3.4 Media Processing Steps

Several crucial processes in the application's workflow are included in media processing, including:

Media Upload: A safe and scalable storage service called Amazon S3 is where media files are originally uploaded.

Processing Trigger: When media is uploaded, an AWS Lambda function is launched to start processing any images or videos. This may entail operations like picture scaling or video encoding.

Processed Media Storage: For effective retrieval, indexing, and administration, processed media files are kept in Amazon S3 while metadata is kept in AWS DynamoDB.

Optional transcoding: AWS Elastic Transcoder may be incorporated for video transcoding and optimisation, ensuring that media material is provided in a variety of formats appropriate for various devices and network conditions.

4. AWS Architecture in Practice

Having delved into the intricacies of the AWS architecture, we can now analyze how this design adeptly addresses the pain points identified in the previously described business scenario. Furthermore, we will explore an alternative solution and discuss the design criteria.

4.1 Alternative Design

During the architectural development process, two variants emerged: the previous design (Appendix C), which mostly used basic AWS services, and the present architecture, which proved to be significantly more streamlined and efficient.

Users accessing media would be directed via CloudFront to an EC2 instance hosting the website in the original design. This website, which was housed in a private subnet, could only be accessed via a NAT gateway . The alternative approach, on the other hand, uses AWS Amplify and the S3 bucket to allow users with direct access to the website's source code, obviating the need for instances and gateways. This upgrade emphasises the architecture's inherent high availability.

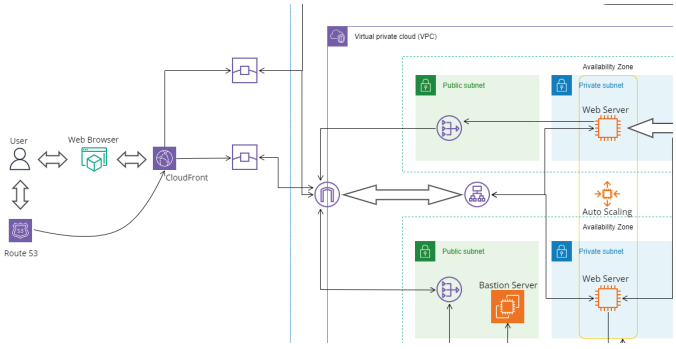

Figure 3: Viewing EC2 Instance

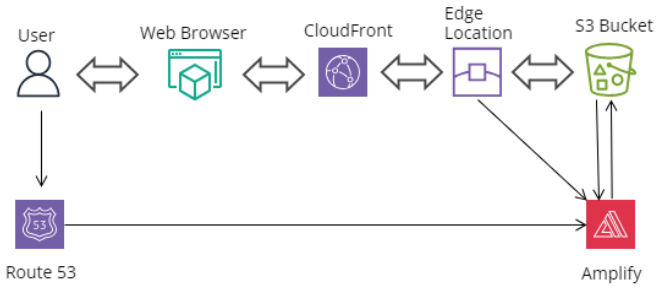

Figure 4: Website AMplify

Uploading material: The original design called for providing original material to an S3 bucket, which would then activate a step function that would process the media and return it to the bucket.

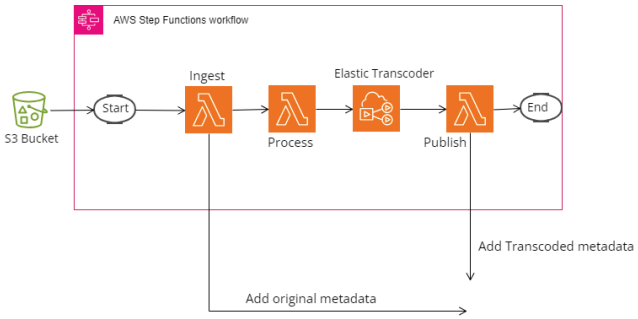

Figure 5: Uploading Media

At the same time, the metadata for the media was transmitted to the database, replicating the procedure for information handled by the AWS Transcoder. In the alternative approach, as media flows through AWS Amplify, an API is triggered to submit a PUT request to the ServiceCategorizer. The media is then directed to the Reliable Fanout procedure, which controls the media type.

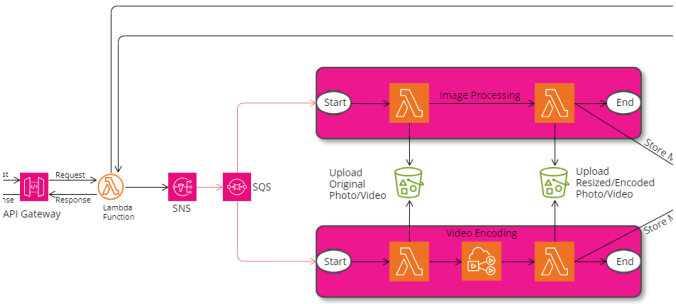

Figure 6: Meida decouple

VPC vs. Serverless Three-Tier Architecture: Adopting a serverless three-tier architecture with automatic scalability and comprehensive security was a deliberate decision. Administrators were responsible for monitoring instances, bastion hosts, and manually configuring security groups or NACLs in the original VPC-based design. With the serverless design, however, AWS Amplify manages scaling and security via authentication, authorisation, data encryption, and IAM roles. While the serverless design has certain infrastructure restrictions, these are minimal in comparison to the benefits of efficiency and security

4.2 Design Rationale

Visual representations of alternative architecture should be included alongside extensive verbal descriptions to improve clarity and comprehension. Visual aids are important in explaining difficult architectural principles.In terms of the VPC vs. serverless contrast, it's worth mentioning that, while serverless architectures provide simplicity and automated scalability, there are some situations where VPCs may be the better choice owing to strong data control needs. Industries controlled by tight rules, such as healthcare, may choose VPCs to ensure robust data management and security, especially when managing sensitive information.

4.3Design Criteria and Solutions

After a thorough assessment of AWS services, the following design criteria are outlined for each pillar of the well-architected framework, along with their respective solutions

Performance

Criterion 1: Low Latency

Solution: Implement content caching at AWS CloudFront edge locations to minimise content delivery latency.

Criterion 2: Optimized Resource Utilization

Solution: AWS Auto Scaling allows you to automatically alter the number of Amazon EC2 instances based on traffic, guaranteeing effective resource utilisation.

Criterion 3: Scalability Planning

Solution: Leverage AWS Application Load Balancers to distribute incoming traffic across multiple instances for efficient scaling.

Criterion 4: Caching Strategies

Solution: Implement in-memory caching for frequently requested data with Amazon ElastiCache to reduce the pressure on backend databases.

Criterion 5: Content Delivery Optimization

Solution: Configure Amazon CloudFront to serve content to users from the closest edge location, resulting in faster content delivery.

Scalability

Criteria 1: Mechanisms for Auto-scaling

Solution: Use AWS Auto Scaling to provide auto-scaling, which automatically adjusts the number of instances depending on user-defined policies.

Critera 2: Load Balancing

Solution: For load balancing, use Amazon Elastic Load Balancers to spread incoming traffic across different instances.

Criteria 3: Database Sharding

Solution: To improve database scaling, use Amazon RDS (Relational Database Service) for automatic database sharding.

Criteria 4: Serverless Architecture

Solution: Adopt a serverless architecture, utilising AWS Lambda for autonomous scalability without operator intervention.

Criterion 5: Horizontal and Vertical Scaling

Solution: Depending on the needs, implement both horizontal scaling (adding new instances) and vertical scaling (raising an instance's resources).

Reliability

Criteria 1: Fault Tolerance

Solution: Use AWS Availability Zones to implement redundancy and failover techniques to achieve fault tolerance.

Criteria 2: Redundancy Strategies

Solution: Use Amazon S3 cross-region replication to maintain data redundancy across many geographic regions.

Criteria 3: Monitoring and notifications
Solution: Configure Amazon CloudWatch for full system monitoring and alarms to get notifications proactively

Criteria 4: Disaster Recovery
Solution: Implement a disaster recovery strategy and set up automatic backups using Amazon RDS snapshots.

Criteria 5: High Availability
Solution: For high availability, use AWS Elastic Load Balancers and Amazon Route 53 to route traffic to healthy instances.

Security

Criteria 1: Authentication and Authorisation
Solution: Use Amazon Cognito to implement robust authentication and role-based access control using AWS Identity and Access Management (IAM).

Criteria 2: Data Encryption
Solution: Use AWS Certificate Manager to encrypt data in transit and Amazon S3 server-side encryption to encrypt data in storage.

Criteria 3: Identity and Access Management (IAM)

Solution: Define IAM roles and rules to control user access securely while giving the least amount of authority required.
Criteria 4: Security Groups and Network ACLs
Solution: Configure network ACLs and security groups to regulate network traffic and safeguard resources from unauthorised access.

Criterion 5: Penetration Testing and Security Audits

Solution: Conduct penetration testing and security audits on a regular basis to detect and address vulnerabilities, assuring the greatest degree of security.

Analysis of Costs

Given the application's exponential expansion, with a doubling rate every 6 months expected for the next 2 to 3 years, it's critical to evaluate the operational expenditures. The following cost breakdowns are offered for various media upload sizes:

1. 50GB Media Upload:

Total Cost for a 50GB Media Upload: $124.54

Breakdown:

Approximately $40.00 for the Content Delivery Network

Web design costs around $55.00.

$29.54 for the media processing procedure

Appendix D has a detailed cost breakdown.

2. 100GB Media Upload:

Forecasted Cost: $308.88

More detailed cost information is available in Appendix E.

3. 200GB Media Upload:

Forecasted Cost: $744.39

Further details are provided in Appendix F.

It's important to note that these cost predictions are based on the application's expected growth. The architecture's scalability and resource management will be crucial in containing these costs as the application expands.

4.4 Compliance with Business Scenarios and Justification for Designed Architecture

- **Infrastructure Administration:**

To reduce the requirement for in-house system management and storage, the design expertly employs AWS managed cloud services such as AWS S3, AWS Amplify, AWS CloudFront, AWS DynamoDB, and AWS Lambda. This strategic deployment not only reduces operational costs but also improves scalability, dependability, and cost-effectiveness, solving the issue of infrastructure management.

- **Scalability**

With the Photo Album application growing biannually, the suggested architecture is rigorously built to respond to projected growth by using a serverless/event-driven approach. Managed cloud services, such as AWS Lambda functions, provide auto-scalability, ensuring that fluctuating traffic loads are handled seamlessly. The smart usage of AWS S3 and AWS DynamoDB increases system capacity and ensures great performance. With AWS SNS, AWS SQS, and AWS Lambda, the event-driven approach decouples application components, resulting in improved scalability and dependability. Without the requirement for on-premises system administration, the design can efficiently control traffic spikes and drops.

- **Database Enhancement**

The choice to switch from AWS RDS to DynamoDB is a critical optimisation step. As a fully managed NoSQL database service, DynamoDB ensures automated resource scaling depending on visitor variations, aligning the application with scalability as well as cost-efficiency. DynamoDB's serverless nature, with billing depending on data saved and throughput utilised, provides a cost-effective database solution. Data replication across several availability zones improves user experience and operational convenience, outperforming AWS RDS's resource-intensive manual scaling and maintenance.

- **Performance Across Regions**

As the popularity and reach of the Photo Album application grow, the architecture's cross-region performance becomes increasingly important. The design dramatically enhances performance by leveraging AWS CloudFront and AWS Amplify. CloudFront's content delivery network optimises content delivery with low latency, and integration with AWS Amplify enables effective cross-region traffic management. This strategic strategy delivers a "edge" advantage, encouraging higher customer happiness, better business outcomes, and a competitive advantage.

- **Evolution and Optimisation of Media Processing**

The web application's expanding functionality and media processing demand optimisation. The architecture employs a fanout architecture, which is a distributed computing structure that effectively processes huge amounts of data in parallel. A serverless/event-driven strategy is used to organise media processing, which includes AWS Lambda services. This method provides autonomous and scalable media processing without the need for human interaction. Furthermore, the architecture is separated with care to support reusability and feature expansion, allowing for easy adaption to new business requirements.

5.Conclusion

Finally, the suggested cloud-based photo album architecture based on AWS services strategically matches with the Photo Album application's dynamic development and needs. It not only tackles the stated pain areas, but it also provides improved scalability, dependability, cost-effectiveness, and performance. The use of serverless/event-driven techniques in the design provides smooth scaling, while effective infrastructure management frees up resources to focus on business development. The move to DynamoDB, along with cross-region speed optimisation, gives a competitive advantage. The growth of media processing via fanout architecture ensures scalability and reusability, opening the door for future feature additions.

We welcome more talks and enquiries about how this design might be changed and customised to meet the specific demands of your company.

References

Amazon Web Services, Inc. (2023). Amazon Simple Queue Service (SQS). Retrieved from https://aws.amazon.com/sqs/

Amazon Web Services, Inc. (2023). AWS Amplify. Retrieved from https://aws.amazon.com/amplify/

Amazon Web Services, Inc. (2023). AWS CloudFront. Retrieved from https://aws.amazon.com/cloudfront/

Amazon Web Services, Inc. (2023). AWS DynamoDB. Retrieved from https://aws.amazon.com/dynamodb/

Amazon Web Services, Inc. (2023). AWS Lambda. Retrieved from https://aws.amazon.com/lambda/

Amazon Web Services, Inc. (2023). AWS Route 53. Retrieved from https://aws.amazon.com/route53/

Amazon Web Services, Inc. (2023). AWS Simple Notification Service (SNS). Retrieved from https://aws.amazon.com/sns/

Amazon Web Services, Inc. (2023). AWS Well-Architected Framework – Performance Pillar. Retrieved from https://d1.awsstatic.com/whitepapers/architecture/AWS-Performance-Efficiency-Pillar.pdf

Amazon Web Services, Inc. (2023). AWS Well-Architected Framework – Scalability Pillar. Retrieved from https://d1.awsstatic.com/whitepapers/architecture/AWS-Scalability-Pillar.pdf

Amazon Web Services, Inc. (2023). AWS Well-Architected Framework – Reliability Pillar. Retrieved fromhttps://d1.awsstatic.com/whitepapers/architecture/AWS-Reliability-Pillar.pdf

Amazon Web Services, Inc. (2023). AWS Well-Architected Framework – Security Pillar. Retrieved from https://d1.awsstatic.com/whitepapers/architecture/AWS-Security-Pillar.pdf

Appendix A: Media Processing Step Procedure

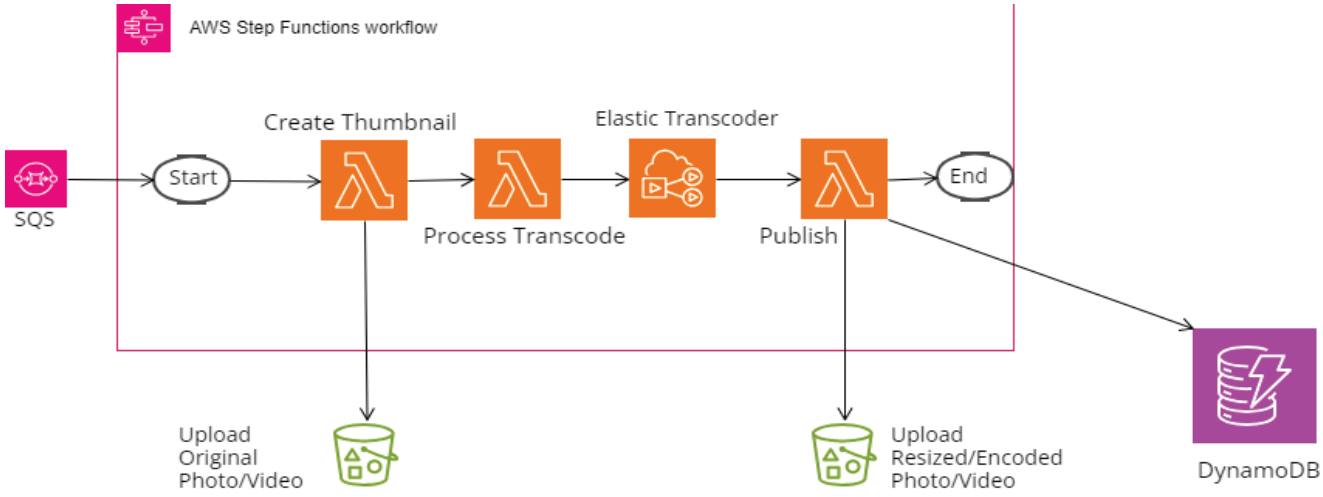

Figure 7: Media Steps

An SQS queue commences the process in this media processing sequence, triggering a series of stages. To begin, AWS Lambda functions generate thumbnails for uploaded material and save them in the original media folder of an Amazon S3 bucket. Following that, another Lambda function prepares the media for transcoding by selecting appropriate formats and creating metadata for the transcoded file. Elastic Transcoder then transcodes the media and delivers it to a publishing Lambda function. The transcoded material is saved in S3 in the TranscodedMedia folder, while the metadata is kept in DynamoDB. This adaptable and reusable approach easily allows the incorporation of new features, making it a solid alternative for changing application requirements.

Appendix B: UML Diagrams for the function and features of the architecture

Viewing UML

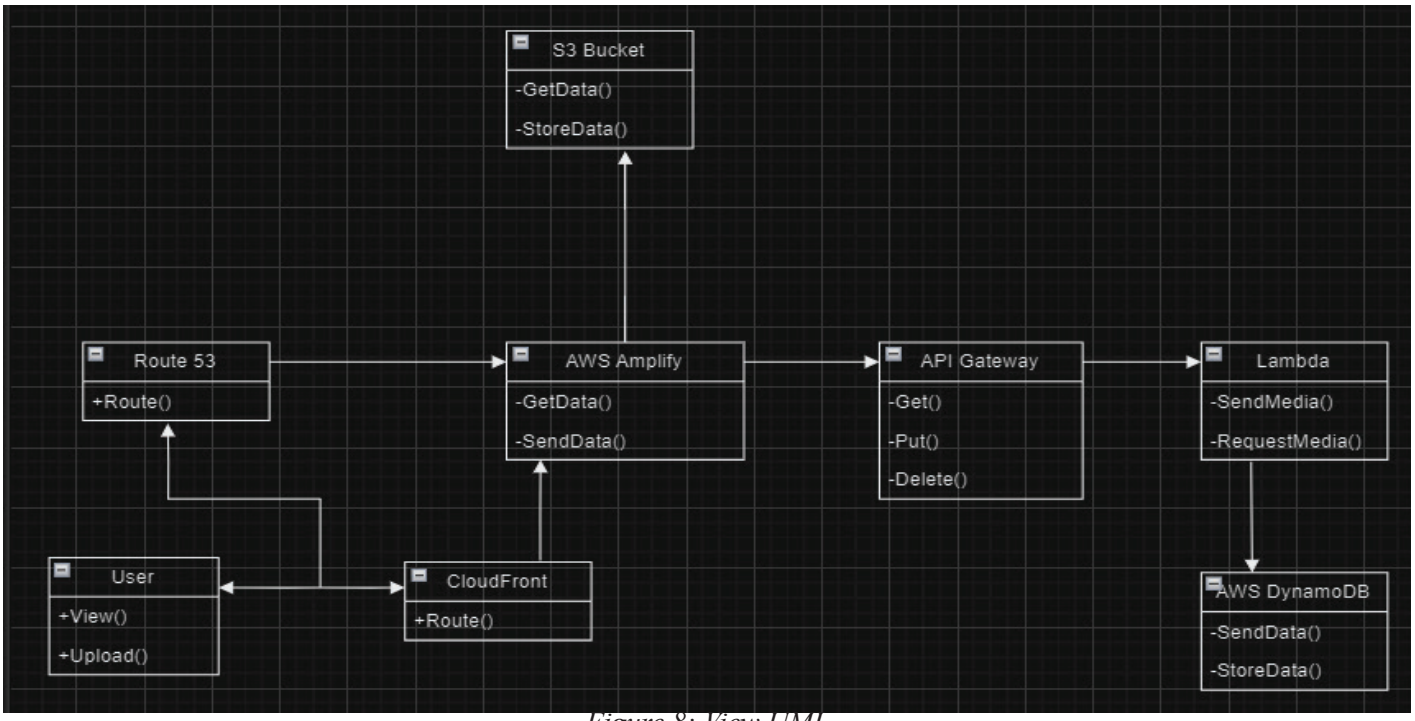

Figure 8: View UML

Users' interactions with the website under the proposed architecture comprise a dynamic routeing mechanism optimised for user experience:

User Website Exploration:

User Request: When visitors visit the website, their requests are efficiently addressed.

CloudFront: AWS CloudFront is the content delivery network that routes users to the closest edge site. This reduces latency and speeds up website loading times.

A**WS Amplify:** When the user arrives at the edge point, AWS Amplify takes control and displays the webpage to them. It is in charge of the presentation layer and guarantees that it is always available.

Uploading UML

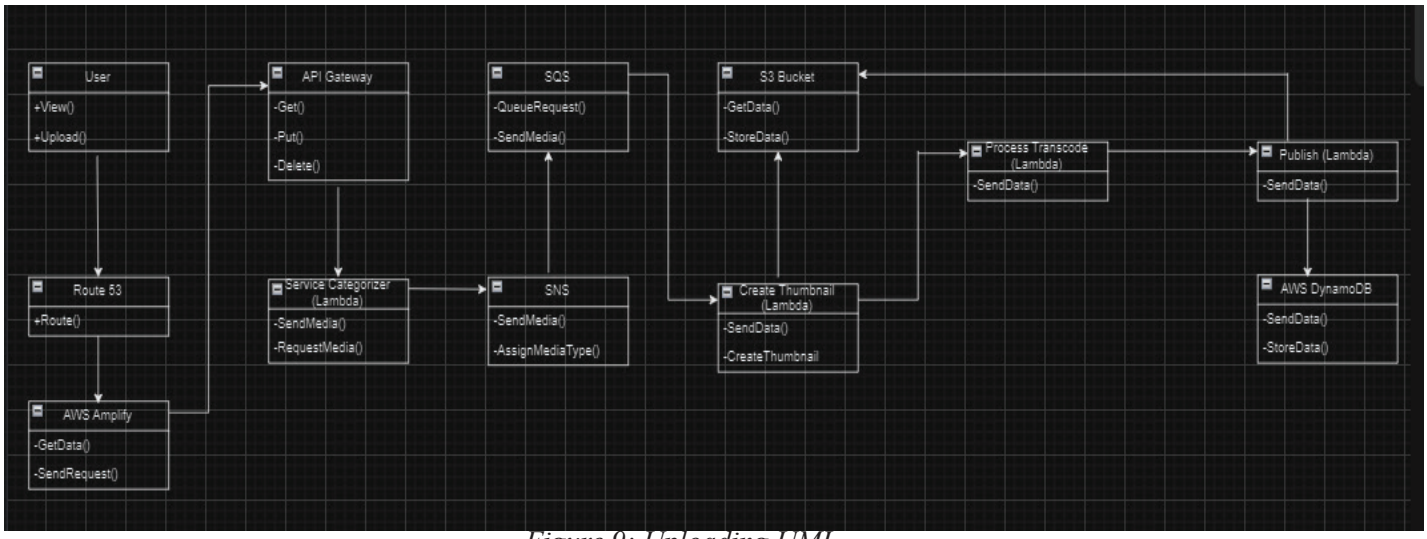

Figure 9: Uploading UML

When a user initiates a media upload, the process begins by sending the media to AWS Amplify over Route 53 and CloudFront. In consequence, AWS Amplify sends an upload request to the API, which activates the PUT mechanism for media submission. Following that, the material is classified using the Service Categorizer function, where it is assigned a subject depending on its media type, such as image_topic or video_topic. This subject serves as an AWS SNS (Simple Notification Service) directive, directing media to the proper AWS SQS (Simple Queue Service) for orderly processing. The original material, together with its thumbnail, is uploaded to an S3 storage bucket via a thumbnail generation function as the process progresses. Concurrently, the material is transcribed using Amazon Elastic Transcoder (AET), which generates the required information. The transcoded material and its associated information are then saved in the S3 bucket and DynamoDB database, delivering a simplified and organised media processing workflow.

Appendix C: Initial version of the architecture

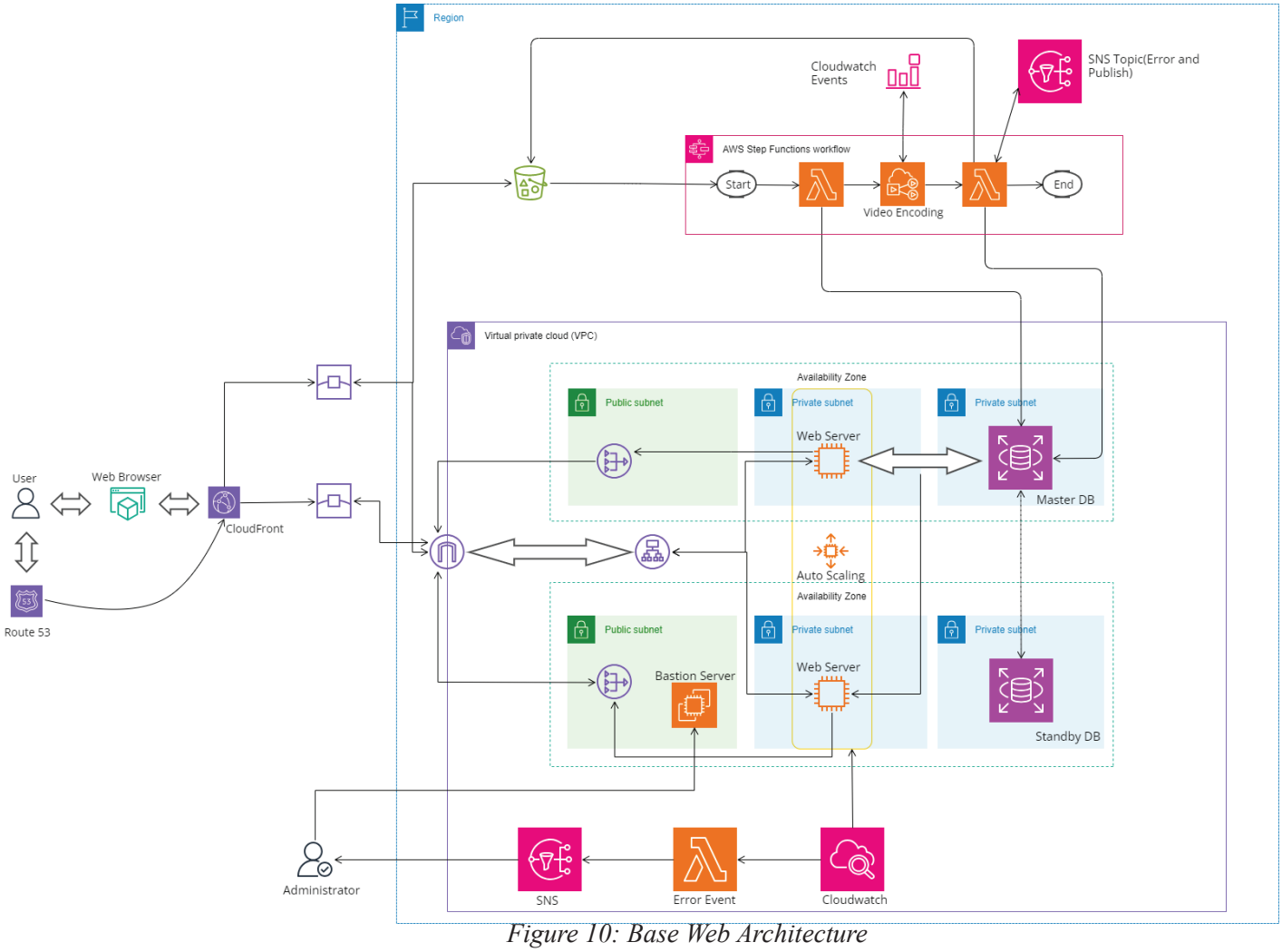

Figure 10: Base Web Architecture

Appendix D: 50GB Meida Upload Cost (Estimated)

Services	Cost of each feature	Total
DynamoDB	• Write Request Unit (WRU): Each WRU = 1 write operation for 1 item with size up to 1kB. 50 x 1024^2 / 1000000 x $1.25 = **$65.536** • Read Request Unit (RRU): Each RRU = 1 read operation for 1 item with size up to 4kB. 50 x 1024^2 / 1000000 / 4 x $0.25 = **$3.2768** • Data Storage (per month): 25 x 0 + 25 x $0.25 = **$6.25** • Data Transfer IN: **$0** • Data Transfer OUT (per month): **$0**	$65.536 + $3.2768 + $6.25 = **$75.0628**
Route 53	Assume all the media uploaded are images and each image's size is 150kB. • Hosted zone (per month): **$0.50** • Standard Queries (per month): 50 x 1024^2 / 150 / 1000000 x $0.40 = **$0.14**	$0.50 + $0.14 = **$0.64**
CloudFront	Assume the price in Australia is the average price. • Data Transfer Out (per month): $0.114 x 50 = **$5.7** • Assume that the average size of images uploaded to the website is 150kB, and all the media uploaded are images. Assume that each image will be associated with 3 HTTPS requests. The price for HTTPS requests will be up to: 50 x 1024^2 / 150 / 10000 x 3 x $0.0125 = **$1.31072**	Up to $5.7 + $1.31072= **$7.01072**
S3	• S3 Standard - Infrequent Access: $0.0125 x 50 = **$0.625** • Data Transfer IN: **$0** • Data Transfer OUT (per month): $0.09 x 50 = **$4.5**	$0.625 + $4.5 = **$5.125**
Amplify	It's free for the first 12 months, so the cost here is **$0**.	**$0**
API Gateway	Assume all the media uploaded are images and each image's size is 150kB. • API Calls (per month): Up to: 50 x 1024^2 / 150 / 1000000 x $3.50 = **$1.2233**	**$1.2233**
SQS	Assume all the media uploaded are images and each image's size is 150kB. • FIFO Queues (per month): **$0** • Data Transfer IN: **$0** • Data Transfer OUT (per month): $0.09 x 50 = **$4.5**	**$4.5**
Total		**$93.56**

Appendix E: 100GB Meida Upload Cost (Estimated)

Services	Cost of each feature	Total
DynamoDB	• Write Request Unit (WRU): Each WRU = 1 write operation for 1 item with size up to 1kB. 100 x 1024^2 / 1000000 x $1.25 = **$131.072** • Read Request Unit (RRU): Each RRU = 1 read operation for 1 item with size up to 4kB. 100 x 1024^2 / 1000000 / 4 x $0.25 = **6.5536** • Data Storage (per month): Up to now, the total of data is about: 50 x 6 + 100 = 400 GB. 25 x $0 + 375 x $0.25 = **$93.75** • Data Transfer IN: **$0** • Data Transfer OUT (per month): **$0**	$131.072 + $6.5536 + $93.75 = **$231.378**
Route 53	Assume all the media uploaded are images and each image's size is 150kB. • Hosted zone (per month): **$0.50** • Standard Queries (per month): 100 x 1024^2 / 150 / 1000000 x $0.40 = **$0.28**	$0.50 + $0.28 = **$0.78**
CloudFront	Assume the price in Australia is the average price. • Data Transfer Out (per month): $0.114 x 100 = **$11.4** • Assume that the average size of images uploaded to the website is 150kB, and all the media uploaded are images. Assume that each image will be associated with 3 HTTPS requests. The price for HTTPS requests will be up to: 100 x 1024^2 / 150 / 10000 x 3 x $0.0125 = **$2.62144**	Up to $11.4 + $2.62144 = **$14.02144**
S3	• S3 Standard - Infrequent Access: $0.0125 x (50 x 6 + 100) = **$5** • Data Transfer IN: **$0** • Data Transfer OUT (per month): $0.09 x 100 = **$9**	$5 + $9 = **$14**
Amplify	It's free for the first 12 months, so the cost here is **$0**.	**$0**
API Gateway	Assume all the media uploaded are images and each image's size is 150kB. • API Calls (per month): Up to: 100 x 1024^2 / 150 / 1000000 x $3.50 = **$2.4466**	**$2.4466**
SQS	Assume all the media uploaded are images and each image's size is 150kB. • FIFO Queues (per month): **$0** • Data Transfer IN: **$0** • Data Transfer OUT (per month): $0.09 x 100 = **$9**	**$9**
Lambda	Assume all the media uploaded are images and each image's size is 150kB; assume that each image is associated with 4 requests. • Requests: 100 x 1024^2 / 150 / 1000000 x 4 x $0.20 = **$0.56** • Memory: **$5.5**	$0.56 + $5.5 = **$6.06**
Total		**$277.69**

Appendix F: 200GB Meida Upload Cost (Estimated)

Services	Cost of each feature	Total
DynamoDB	• Write Request Unit (WRU): Each WRU = 1 write operation for 1 item with size up to 1kB. 200 x 1024^2 / 1000000 x \$1.25 = **\$262.144** • Read Request Unit (RRU): Each RRU = 1 read operation for 1 item with size up to 4kB. 200 x 1024^2 / 1000000 / 4 x \$0.25 = **13.1072** • Data Storage (per month): Up to now, the total of data is about: 50 x 6 + 100 x 6 + 200 = 1100 GB. 25 x \$0 + 1075 x \$0.25 = **\$268.75** • Data Transfer IN: **\$0** • Data Transfer OUT (per month): **\$0**	\$262.144+ \$13.1072 + \$268.75 = **\$544.0012**
Route 53	Assume all the media uploaded are images and each image's size is 150kB. • Hosted zone (per month): **\$0.50** • Standard Queries (per month): 200 x 1024^2 / 150 / 1000000 x \$0.40 = **\$0.56**	\$0.50 + \$0.56 = **\$1.06**
CloudFront	Assume the price in Australia is the average price. • Data Transfer Out (per month): \$0.114 x 200 = **\$22.8** • Assume that the average size of images uploaded to the website is 150kB, and all the media uploaded are images. Assume that each image will be associated with 3 HTTPS requests. The price for HTTPS requests will be up to: 200 x 1024^2 / 150 / 10000 x 3 x \$0.0125 = **\$5.24288**	Up to \$22.8 + \$5.24288 = **\$28.04288**
S3	• S3 Standard - Infrequent Access: \$0.0125 x (50 x 6 + 100 x 6 + 200) = **\$13.75** • Data Transfer IN: **\$0** • Data Transfer OUT (per month): \$0.09 x 200 = **\$18**	\$13.75 + \$18 = **\$31.75**
Amplify	Assume web app size = 100MB, average size of page requested = 1.5 MB Assume that average build time = 3 minutes each day. Assume that daily active users = 10000. • Build & Deploy: 3 x 30 x 0.01 = **\$0.9** • Data storage: 100 / 1024 x \$0.023 = \$0.00225 • Data Transfer Out: 10000 x (1.5 / 1024) x 30 x \$0.15 = **\$65.918**	\$0.9 + \$65.918 = **\$66.818**
API Gateway	Assume all the media uploaded are images and each image's size is 150kB. • API Calls (per month): Up to: 200 x 1024^2 / 150 / 1000000 x \$3.50 = **\$4.8932**	**\$4.8932**
SQS	Assume all the media uploaded are images and each image's size is 150kB. • FIFO Queues (per month): **\$0** • Data Transfer IN: **\$0** • Data Transfer OUT (per month): \$0.09 x 200 = **\$18**	**\$18**
Lambda	Assume all the media uploaded are images and each image's size is 150kB; assume that each image is associated with 4 requests. • Requests: 200 x 1024^2 / 150 / 1000000 x 4 x \$0.20 = **\$1.12** • Memory: **\$5.5**	\$1.12 + \$5.5 = **\$6.62**
Total		**\$701.185**