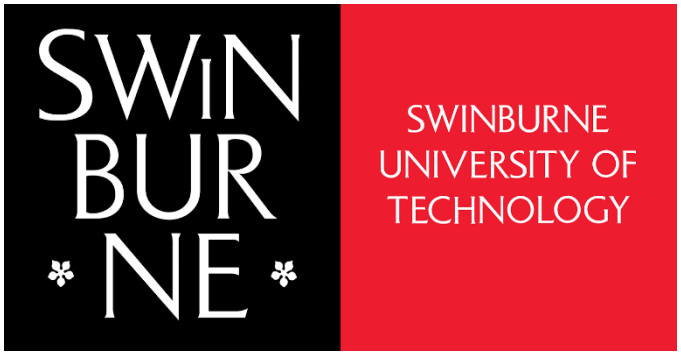

COS-20019/Cloud Computing Architecture

Assignment-3 Report.

Authors: Yu Feng Chew - ID: 103984220.

Dave Nguyen – ID: 104697710.

Jack Edwards – ID: 105179358.

Lab - Monday 4:30 - 6:30, Room: BA411.

Date of submission: 25/05/2025.

Table of Contents

1. Summary

This report presents a serverless, event-driven architectural design for the Photo Album application to address scalability, performance, and cost-efficiency requirements. The proposed solution leverages AWS managed services to minimise administrative overhead while ensuring global availability and automatic scaling to handle growing demand.

2. Architectural Diagram

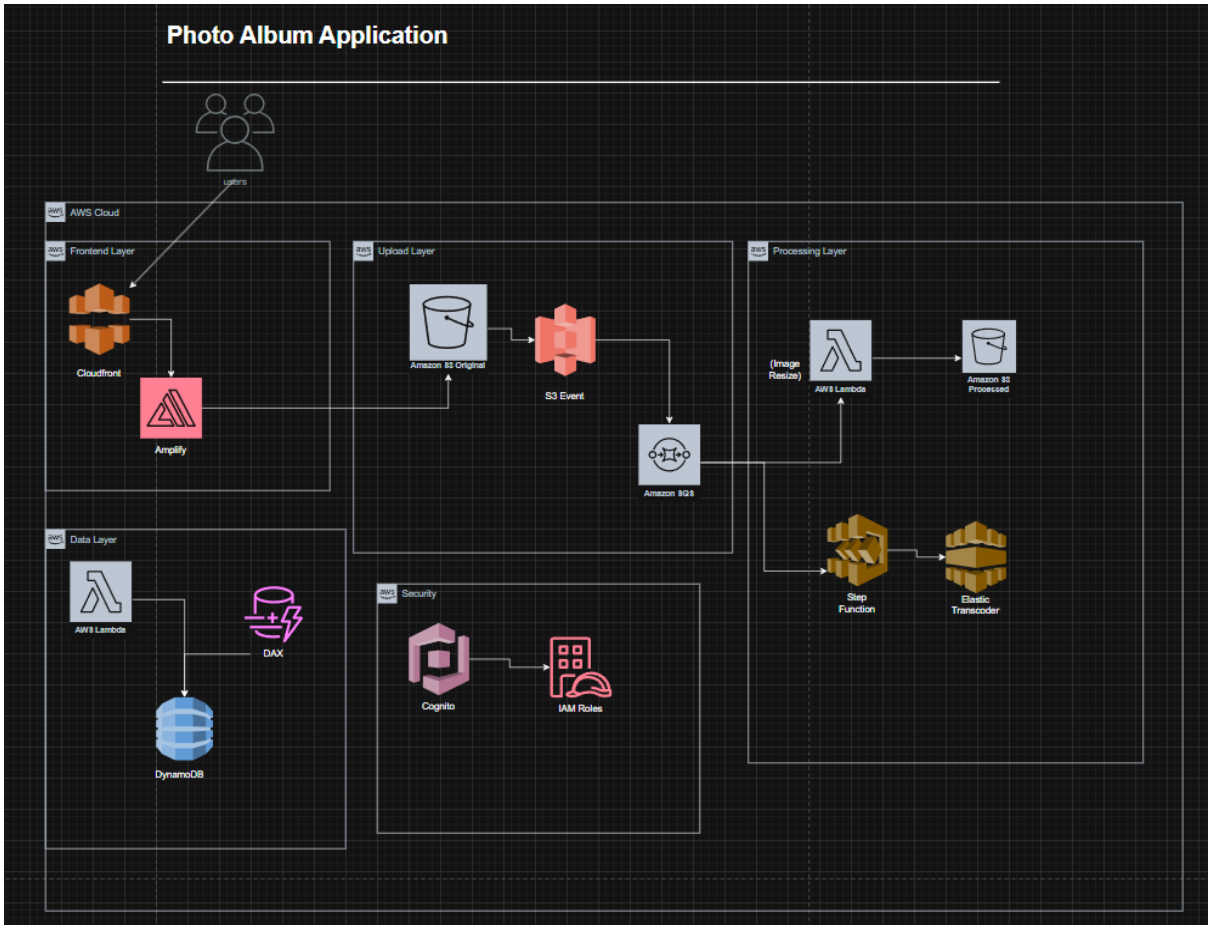

Fig.1: Architectural Diagram.

Key components:

- Amazon CloudFront for global content delivery.
- AWS Lambda for serverless compute.
- Amazon S3 for media storage.
- Amazon DynamoDB for the database.
- Amazon SQS/SNS for event-driven processing.
- AWS Step Functions for workflow orchestration.

- Amazon Recognition for future AI capabilities.

3. Server Descriptions & Justifications

Content Delivery and Frontend

Amazon CloudFront:

- Distributes content globally with low latency.
- Integrates with S3 for media delivery.
- Provides caching to reduce backend load.
- Alternative: Could use Akamai, but AWS integration is simpler.

AWS Lambda + API Gateway:

- Handles application logic in serverless functions.
- Scales automatically with demand.
- Cost-effective for variable workloads.
- Alternative: EC2 would require capacity planning and scaling configuration.

2. Data Storage

Amazon S3:

- Stores original and processed media files.
- Provides 99% durability.
- Versioning is enabled for backup.
- Alternative: EBS volumes would be more expensive and less scalable.

Amazon DynamoDB:

- NoSQL database for metadata and user data.
- Scales automatically with demand.
- Lower cost than RDS for simple data structures.
- Alternative: Aurora Serverless could be considered, but more expensive.

3. Media Processing Pipeline

Amazon S3 Event Notifications:

- Triggers processing when new media is uploaded.
- Alternative: Could use CloudWatch Events, but S3 integration is simpler.

Amazon SQS:

- Decouples media upload from processing.

- Allows different worker types for different tasks.

AWS Lambda/EC2/Fargate:

Mixed processing approach:

- Lambda for simple transformations (thumbnails).
- EC2/Fargate for complex tasks (video transcoding).
- Alternative: Pure Lambda might hit resource limits for video.

4. Monitoring and Operations

Amazon CloudWatch:

- Monitors all services.
- Triggers alerts for issues.
- Alternative: Third-party tools like Datadog, but more expensive.

Design Rationale

Business Scenario Fulfilment:

1. **Managed services**: The Entire solution uses AWS managed services.
2. **Scalability**: All components scale automatically.
3. **Compute capacity**: Serverless avoids EC2 capacity issues.
4. **Serverless**: Lambda and other serverless services are used throughout.
5. **Database**: DynamoDB replaces relational databases.
6. **Global performance**: CloudFront improves worldwide response.
7. **Future video**: Architecture supports video processing.
8. **Media processing**: Event-driven pipeline with SQS/SNS.

5. Alternative Solutions Analysis:

1. **Compute Options**:
    - EC2: Requires capacity planning, less cost-effective for spiky loads
    - ECS/EKS: More complex than serverless for this use case

- **Chosen**: Lambda for most tasks, EC2 only for heavy video.

2. **Database Options**:
    - RDS: More expensive, overkill for simple data.
    - Aurora Serverless: Good but higher cost.
    - **Chosen**: DynamoDB for simplicity and scale.
3. **Messaging Options**:
    - SNS: The Push model is less flexible for processing.
    - Kinesis: Overkill for this volume.
    - **Chosen**: SQS for decoupled processing.
4. **Architecture Tiers**:
    - A traditional 3-tier would require more management.
    - **Chosen**: Serverless microservices for flexibility.

6. Design Criteria Evaluation:

1. **Performance**:
    - CloudFront provides global low-latency.
    - Lambda scales instantly to demand.
    - Processing decoupled via SQS.
2. **Scalability**:
    - All components scale automatically.
    - New processors can be added easily.
    - Database scales seamlessly.
3. **Reliability**:
    - Multi-AZ deployment.
    - S3 provides durability.
    - SQS ensures no lost messages.
4. **Security**:

- IAM for fine-grained access control.
- Data encrypted at rest and in transit.
- VPC for compute resources.

5. **Cost**:
    - Pay-per-use model for most services.
    - No idle resource costs.

7. Conclusion

The proposed serverless/event-driven architecture meets all current requirements while providing flexibility for future growth. By leveraging AWS managed services, we minimise administrative overhead while ensuring scalability, reliability, and global performance. The decoupled design allows for easy extension of processing capabilities as needs evolve.