Project-1: Serverless Image Processing with AWS Lambda, Amazon S3, and DynamoDB

1. Introduction

1.1 Purpose

The purpose of this document is to provide a detailed Software Requirement Specification (SRS) for the development of a serverless image processing application. The application will use AWS Lambda, Amazon S3, and DynamoDB to process images, store metadata, and provide a user interface for viewing processed images.

1.2 Scope

This project involves creating an application that processes images uploaded to an S3 bucket, adds a watermark, and saves the processed images back to S3. It also involves storing metadata about the processed images in DynamoDB, exposing REST APIs to access this metadata, and developing a React web application for displaying the processed images.

1.3 Definitions, Acronyms, and Abbreviations

- **AWS**: Amazon Web Services
- **S3**: Simple Storage Service
- **DynamoDB**: NoSQL database service provided by AWS
- **API**: Application Programming Interface
- **Lambda**: AWS service that runs code in response to events
- **SRS**: Software Requirement Specification

1.4 References

- AWS Lambda documentation
- Amazon S3 documentation
- Amazon DynamoDB documentation
- AWS API Gateway documentation
- AWS Secrets Manager documentation
- AWS CodeBuild documentation
- SonarQube documentation
- React documentation
- Amazon Cognito documentation

1.5 Overview

This document describes the requirements for the serverless image processing application, including system features, external interface requirements, system architecture, and quality attributes.

2. Overall Description

2.1 Product Perspective

The application will be a new, standalone system. It will utilize various AWS services to achieve serverless image processing, metadata storage, and user interface functionalities.

2.2 Product Functions

- Upload images to S3
- Automatically process images to add a watermark using AWS Lambda
- Store processed images in S3
- Save metadata about processed images in DynamoDB
- Expose REST APIs to retrieve metadata and image URLs
- Provide a React web application to view processed images and metadata
- Secure web application with Amazon Cognito

2.3 User Classes and Characteristics

- **End Users**: Users who upload images and view processed images through the web application.
- **Administrators**: Users who manage application settings and monitor performance.

2.4 Operating Environment

- AWS cloud environment
- Modern web browsers for the React web application

2.5 Design and Implementation Constraints

- Use Java for AWS Lambda functions
- Use React for the web application
- Use AWS services for deployment and security

2.6 Assumptions and Dependencies

- Users have an AWS account
- Users have basic knowledge of AWS services
- Internet connectivity for accessing AWS and the web application

3. System Features

3.1 Image Upload and Processing

Description: Users upload images to an S3 bucket, which triggers a Lambda function to add a watermark and save the processed image back to S3.

- **Functional Requirements:**
  - FR1: Create an S3 bucket to store original images.
  - FR2: Implement a Lambda function to add a watermark to images.
  - FR3: Save the processed images to a different S3 bucket or folder.

3.2 Metadata Storage

Description: Store metadata about processed images in DynamoDB.

- **Functional Requirements:**
  - FR4: Create a DynamoDB table to store image metadata.
  - FR5: Store information such as image URL, upload timestamp, and watermark details.

3.3 REST APIs

Description: Provide REST APIs to retrieve processed image metadata and URLs.

- **Functional Requirements:**
  - FR6: Use AWS API Gateway to expose REST APIs.
  - FR7: Implement API endpoints to retrieve image metadata.

3.4 Web Application

Description: Develop a React web application to display processed images and their metadata.

- **Functional Requirements:**
  - FR8: Fetch metadata and image URLs using REST APIs.
  - FR9: Display the processed images and metadata in the web application.

3.5 Security

Description: Secure the web application using Amazon Cognito.

- **Functional Requirements:**
  - FR10: Implement user authentication using Amazon Cognito.
  - FR11: Restrict access to authenticated users only.

3.6 Build and Code Quality

Description: Implement automated builds and ensure code quality.

- **Functional Requirements:**
  - FR12: Use AWS CodeBuild for automated builds.
  - FR13: Integrate SonarQube to ensure code quality during the build process.

4. External Interface Requirements

4.1 User Interfaces

- React web application interface for uploading images and viewing processed images.

4.2 Software Interfaces

- AWS SDK for Java
- React library

4.4 Communication Interfaces

- REST APIs for interaction between the web application and backend services.

5. System Architecture

5.1 AWS Services Architecture

- **Amazon S3**: For storing original and processed images.
- **AWS Lambda**: For image processing.
- **Amazon DynamoDB**: For storing metadata.
- **AWS API Gateway**: For exposing REST APIs.
- **AWS Secrets Manager**: For storing sensitive information like watermark details.
- **AWS CodeBuild**: For automated builds.
- **Amazon Cognito**: For user authentication.

5.2 Data Flow

1. User uploads an image to S3.
2. S3 triggers a Lambda function to process the image.
3. Processed image is saved back to S3.
4. Lambda function stores metadata in DynamoDB.
5. API Gateway provides endpoints to retrieve metadata.
6. React web application fetches and displays the processed images and metadata.

6. Quality Attributes

6.1 Performance

- Lambda functions should process images within acceptable time limits.
- APIs should respond quickly to metadata requests.

6.2 Scalability

- The application should handle multiple concurrent image uploads and processing requests.

6.3 Security

- Use Amazon Cognito for secure user authentication.
- Use AWS Secrets Manager to securely manage sensitive information.

6.4 Maintainability

- Use modular code to facilitate maintenance and updates.
- Implement unit tests using jUnit for Lambda functions.

AWS Cloud Activities

1. Infrastructure Setup and Management

- **S3 Buckets**: Create and configure S3 buckets for storing original and processed images.
- **DynamoDB**: Create and configure the DynamoDB table for storing image metadata.
- **Lambda Functions**: Set up the AWS Lambda environment, including permissions and triggers.
- **API Gateway**: Set up and configure API Gateway to expose REST APIs for metadata retrieval.
- **Secrets Manager**: Store sensitive information like watermark details in AWS Secrets Manager and manage access permissions.
- **CodeBuild**: Set up AWS CodeBuild for continuous integration and deployment.

2. Security and Compliance

- **IAM Roles and Policies**: Define IAM roles and policies for Lambda functions, S3, DynamoDB, and other AWS services.
- **Cognito**: Set up Amazon Cognito for user authentication and access control for the web application.

3. Monitoring and Logging

- **CloudWatch**: Configure Amazon CloudWatch for logging and monitoring Lambda functions and other AWS services.
- **Alerts**: Set up CloudWatch alarms for critical metrics and incidents.

Development Activities

1. Web Application Development

- **React Application**: Develop the React web application to display processed images and their metadata.
- **API Integration**: Integrate the web application with the REST APIs exposed by the AWS backend.
- **User Interface Design**: Design and implement the user interface, ensuring a responsive and user-friendly experience.

2. Lambda Function Development

- **Java Lambda Functions**: Develop the Java code for the Lambda functions responsible for image processing and interaction with DynamoDB.
- **jUnit Tests**: Write and maintain jUnit tests for the Lambda functions to ensure code quality and functionality.

3. Frontend-Backend Integration

- **API Gateway**: Ensure the React application correctly communicates with the API Gateway endpoints.
- **Data Handling**: Handle the retrieval and display of metadata and image URLs within the web application.