Faical recognition on AWS

by

Kaung Htet Cho

A Project Submitted in Partial Fulfillment of the Requirements for the Cloud computing course

Examination Committee:	Prof. A (Chantri Polprasert)
Nationality:	Myanmar
Degree:	Mechatronics and Machine Intelligence
	ISE, AIT
	Thailand

`https://github.com/KaungHtetCho-22/Face-recognition-on-AWS`

Asian Institute of Technology
School of Engineering and Technology
Thailand
March 2024

AUTHOR'S DECLARATION

I, Kaung Htet Cho, declare that the research work carried out for this project was in accordance with the regulations of the Asian Institute of Technology. The work presented in it are my own and has been generated by me as the result of my own original research, and if external sources were used, such sources have been cited. It is original and has not been submitted to any other institution to obtain another degree or qualification. This is a true copy of the project including final revisions.

Date: March 7, 2024

Name: Kaung Htet Cho

ACKNOWLEDGEMENTS

I would like to thank Professor Chantri Polprasert for for his invaluable guidance and support throughout the development of this project.

ABSTRACT

This project develops a facial recognition-based access control system using Amazon Web Services (AWS) Rekognition to enhance security measures by accurately identifying and verifying employees' identities in real-time. The system employs an efficient registration process, robust authentication mechanism, and seamless integration with an EC2 server backend using the Flask framework. Leveraging various AWS services, such as S3, Lambda, and Key Management Service (KMS), the architecture ensures scalability, security, and cost-effectiveness. The project adheres to the AWS Well-Architected Framework, ensuring the system's reliability, performance efficiency, and sustainability. Rigorous testing demonstrates the solution's accuracy and robustness in real-world scenarios. By successfully implementing this system, the project showcases the potential of cloud computing and deep learning in enhancing security measures and streamlining access control processes, offering organizations a reliable and efficient alternative to traditional methods.

CONTENTS

CHAPTER 1
INTRODUCTION

1.1 Background of the Study

In today's digital landscape, secure access to corporate premises is crucial, yet traditional methods like swipe cards or PIN codes often lack robust security. Facial recognition technology, leveraging biometric authentication, offers a promising solution. With advancements in machine learning, facial recognition systems have become more accurate. Amazon Web Services (AWS) Rekognition provides powerful facial recognition capabilities, enabling easy integration into applications. This study proposes developing a facial recognition-based access control system using AWS Rekognition to accurately identify and verify employees' identities in real-time, streamlining access control processes while enhancing security measures and user experience.

1.2 Statement of the Problem

The problem addressed in this project is the need for a secure and efficient access control system for corporate premises. Traditional access control methods like swipe cards or PIN codes often lack robust security and are prone to unauthorized access and security breaches. Additionally, manual verification processes can be time-consuming and error-prone. There is a growing demand for advanced technologies that can enhance security measures while streamlining access control processes. In this context, the problem is how to develop a facial recognition-based access control system using Amazon Web Services (AWS) Rekognition that accurately identifies and verifies employees' identities in real-time, improving security measures and user experience while addressing the limitations of traditional access control methods.

1.3 Objectives

The primary objective of this project is to develop a robust and efficient facial recognition-based access control system using Amazon Web Services (AWS) Rekognition. The system aims to accurately identify and verify employees' identities in real-time, enhancing security measures and streamlining access control processes. The specific objectives of the project are:

- Collect and preprocess a diverse dataset of employees' facial images to train and test the facial recognition model.

- Evaluate and select the most suitable facial recognition model from AWS Rekognition, considering factors such as accuracy, speed, and scalability.
- Implement a secure and user-friendly registration process to enroll employees' facial images into the system, utilizing AWS Rekognition's APIs to extract facial features and generate unique identifiers (Rekognition IDs).
- Develop a robust authentication mechanism for real-time facial recognition during access attempts, leveraging AWS Lambda functions to trigger face matching capabilities and compare captured facial images with the stored database.
- Integrate the facial recognition system with an EC2 server backend using Flask to provide a seamless and intuitive user interface for registration, authentication, and access control functionalities.
- Ensure the security of sensitive data by implementing encryption using AWS Key Management Service (KMS) for facial images stored in S3.
- Optimize the system's performance, reliability, and cost-effectiveness by leveraging AWS services and adhering to best practices outlined in the Well-Architected Framework.

By achieving these objectives, the project aims to deliver a comprehensive and reliable facial recognition-based access control system that enhances security measures, improves user experience, and demonstrates the power of cloud computing in solving complex problems.

CHAPTER 2
Literature Review

Facial recognition technology has garnered significant attention in recent years due to its wide-ranging applications in security, surveillance, and biometric authentication systems. Several studies have explored the effectiveness and potential challenges associated with implementing facial recognition solutions, particularly those leveraging cloud-based services like Amazon Web Services (AWS) Rekognition.

2.1 Facial Recognition

Facial recognition, essential in various domains like security and authentication, has seen significant advancements with the adoption of deep learning techniques, particularly Convolutional Neural Networks (CNNs). These CNNs excel in learning intricate facial features from images, enabling accurate recognition across diverse conditions. Deep learning models automatically extract relevant features from data, eliminating the need for manual feature engineering and improving adaptability to different environments. Various architectures like Residual Networks (ResNets) and Inception Networks have been proposed, achieving state-of-the-art performance on benchmark datasets. Recurrent Neural Networks (RNNs) and Generative Adversarial Networks (GANs) also contribute to tasks like facial expression recognition and image generation. Despite progress, challenges such as data privacy and bias persist, underscoring the ongoing need for research and innovation in this field. Hu et al. (2015) Onyema et al. (2021)

2.2 AWS Rekognition

AWS Rekognition, a cloud-based image and video analysis service provided by Amazon Web Services (AWS), plays a significant role in advancing facial recognition technology. By leveraging AWS Rekognition, developers gain access to powerful and scalable deep learning models specifically designed for tasks like face detection, face analysis, and face comparison. The service offers a comprehensive suite of features, including real-time face detection and recognition, age and gender estimation, emotion detection, and facial landmark detection. Moreover, AWS Rekognition provides high accuracy and reliability, even in challenging conditions such as low light or occlusions. With its ease of integration, developers can quickly incorporate facial recognition capabilities into their applications without the need for extensive expertise in machine learning or computer vision. Additionally, AWS Rekognition's robust APIs and SDKs allow for seamless integration with other AWS services, enabling developers to build sophisticated and scalable facial recognition systems with ease. Overall, AWS Rekognition enhances facial recognition technology by providing developers with access to state-of-the-art deep learning models, scalability, reliability, and ease of integration, thereby accelerating the development and deployment of facial recognition applications. Rafael, Kusuma, et al. (2020)

2.2.1 AWS Rekognition - index-faces API

Amazon Rekognition's index-faces API plays a crucial role in the underlying process of facial recognition, utilizing advanced deep learning techniques to extract and index facial features from images. While specific details of the algorithm are proprietary to Amazon, the index-faces API is built on state-of-the-art convolutional neural networks (CNNs) that are trained on vast datasets of facial images. These CNNs are capable of detecting key facial landmarks, capturing facial attributes such as facial expressions, and encoding unique facial features into compact representations known as embeddings.

The index-faces API typically follows a multi-step process, starting with face detection, where the algorithm identifies and localizes faces within an image using sophisticated object detection techniques. Once faces are detected, the algorithm extracts discriminative features from each face region, such as the arrangement of facial landmarks, texture patterns, and color information. These features are then transformed into high-dimensional embedding vectors using neural network architectures optimized for feature extraction tasks.

Next, the extracted embeddings are indexed and stored in a database, enabling efficient retrieval and comparison during the identification process. The indexing process involves encoding each facial embedding with a unique identifier, allowing for fast and accurate retrieval of matching faces from the database. During face identification, the algorithm compares query embeddings with the indexed embeddings using similarity metrics such as cosine similarity or Euclidean distance. Matches are then ranked based on their similarity scores, with higher scores indicating closer matches.

Amazon Rekognition's index-faces API leverages the scalability and computational power of cloud infrastructure to process large volumes of images efficiently, making it suitable for a wide range of applications, including security, authentication, and content moderation. While the exact implementation details of the index-faces API are proprietary, its underlying process is grounded in cutting-edge deep learning techniques, enabling robust and accurate facial recognition capabilities. Leonor Estévez Dorantes, Bertani Hernández, León Reyes, and Elena Miranda Medina (2022)

CHAPTER 3
METHODOLOGY

3.1 Methodology

The development of the facial recognition-based access control system using AWS Rekognition involves a systematic approach that encompasses data collection, model selection, system design, and integration. The following steps outline the methodology employed in this project:

3.1.1 Data Collection and Preprocessing

- Gather a diverse dataset of facial images of company employees, ensuring variations in facial expressions, angles, and lighting conditions.
- Preprocess the collected images to ensure consistency and compatibility with the facial recognition model.

3.1.2 Model Selection and Integration

- Evaluate and compare various facial recognition models available in AWS Rekognition based on factors such as accuracy, speed, and scalability.
- Select the IndexFaces API in AWS Rekognition for its automated face printing capabilities, high accuracy, scalability, ease of integration, and cost-effectiveness.
- Integrate the SearchFacesByImage API from AWS Rekognition to efficiently find and retrieve indexed faces from DynamoDB, leveraging its seamless integration with other AWS services.

3.1.3 Registration Process

- Design and implement a user-friendly registration mechanism to enroll people' facial images into the system securely.
- IUtilize AWS Key Management Service (KMS) to encrypt the facial images before storing them in an S3 bucket, ensuring data security.
- Employ AWS Rekognition's APIs to extract facial features and generate unique identifiers (Rekognition IDs) for each registered person.
- Store the generated Rekognition IDs along with the corresponding person names in a DynamoDB table for efficient retrieval during the authentication process.

3.1.4 Authentication Mechanism

- Develop a robust authentication mechanism that enables real-time facial recognition during access attempts.
- Configure AWS Lambda functions to trigger AWS Rekognition's face matching capabilities, comparing captured facial images with the stored database of registered people.
- Implement logic to grant access if a successful match is found or deny access if no match is detected, ensuring the system's security and reliability.

3.1.5 Integration with EC2 server

- IIntegrate the facial recognition system with an EC2 server backend using the Flask framework to provide a seamless and intuitive user interface.
- Design and implement a user-friendly web application that handles registration, authentication, and access control functionalities, ensuring smooth communication between the frontend and backend components.
- Optimize the system's performance and scalability by leveraging the capabilities of the EC2 server and configuring appropriate resources based on the expected user load.

By following this methodology, the project aims to develop a robust, secure, and efficient facial recognition-based access control system that harnesses the power of AWS Rekognition and other AWS services to enhance security measures and streamline access control processes for the organization.

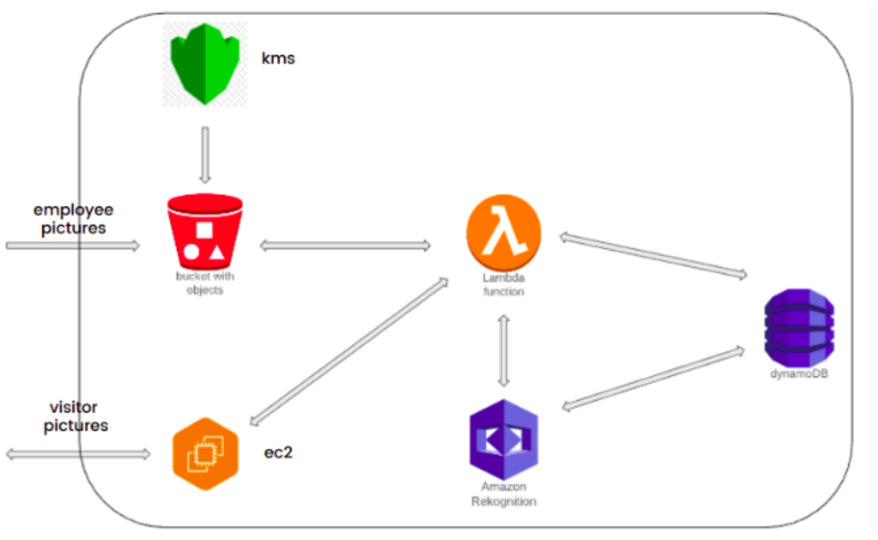


Figure 3.1

Overall Architecture of the System

CHAPTER 4
WELL-ARCHITECTED FRAMEWORK

4.1 Security foundations

- Implement robust identity and access management (IAM) policies to control access to AWS resources used in the project, following the principle of least privilege.
- Encrypt data in transit and at rest, including facial images and other sensitive data, using AWS Key Management Service.

4.2 Operational excellence

- Utilize AWS CloudFormation templates to automate the deployment of the AWS infrastructure, ensuring consistency, repeatability, and reducing the risk of human error during deployments.

4.3 Reliability

- The facial recognition system ensures high availability by utilizing AWS services like Amazon S3, which provides 11-9s durability and 99.99 availability for storing facial images. Ensure that the system is available and accessible to users at all times, allowing them to upload images and receive real-time recognition results without experiencing downtime or service interruptions
- Deploying the system across multiple Availability Zones further minimizes the impact of any single point of failure.

4.4 Performance efficiency

- Optimize Lambda functions by reducing execution time, minimizing dependencies, and maximizing concurrency settings to ensure efficient processing of image uploads and recognition tasks.
- Implement S3 storage optimization by utilizing lifecycle policies and storage classes to automatically move infrequently accessed images to cheaper storage tiers, reducing costs while maintaining accessibility.

4.5 Cost optimization

- mplement S3 lifecycle policies to automatically transition infrequently accessed objects to lower-cost storage classes, such as S3 Glacier or S3 Glacier Deep Archive, reducing storage costs while maintaining data accessibility.

- Right-size EC2 instances, Lambda functions, and DynamoDB tables to match the workload demands, avoiding over-provisioning and under-utilization of resources.

4.6 Sustainability

- Utilize a serverless architecture with AWS Lambda for image processing tasks, eliminating the need for maintaining and provisioning servers, reducing resource wastage and energy consumption associated with idle servers.
- Implement on-demand provisioning by analyzing resource usage patterns and right-sizing EC2 instances, Lambda functions, and DynamoDB tables to match the workload demands, avoiding over-provisioning and under-utilization of resources.

REFERENCES

Hu, G., Yang, Y., Yi, D., Kittler, J., Christmas, W., Li, S. Z., & Hospedales, T. (2015, December). When face recognition meets with deep learning: An evaluation of convolutional neural networks for face recognition. In *Proceedings of the ieee international conference on computer vision (iccv) workshops.*

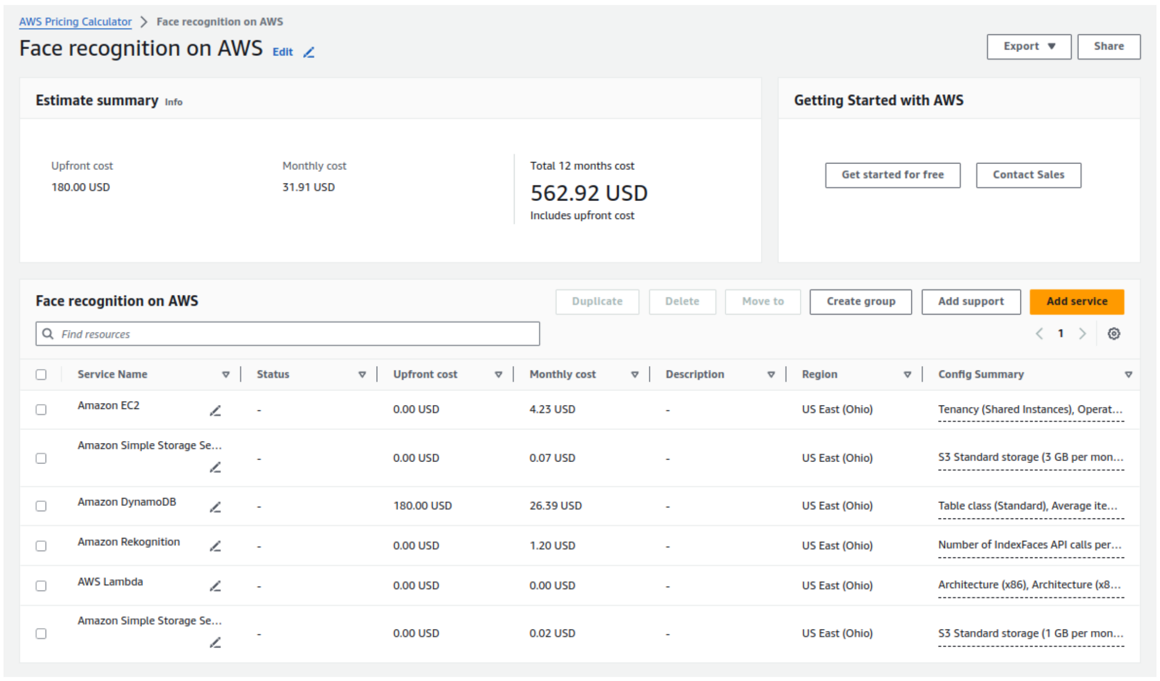
Leonor Estévez Dorantes, T., Bertani Hernández, D., León Reyes, A., & Elena Miranda Medina, C. (2022). Development of a powerful facial recognition system through an api using esp32-cam and amazon rekognition service as tools offered by industry 5.0. In *2022 the 5th international conference on machine vision and applications (icmva)* (pp. 76–81).

Onyema, E. M., Shukla, P. K., Dalal, S., Mathur, M. N., Zakariah, M., Tiwari, B., et al. (2021). Enhancement of patient facial recognition through deep learning algorithm: Convnet. *Journal of Healthcare Engineering*, *2021.*

Rafael, G., Kusuma, H., et al. (2020). The utilization of cloud computing for facial expression recognition using amazon web services. In *2020 international conference on computer engineering, network, and intelligent multimedia (cenim)* (pp. 366–370).

APPENDICES

APPENDIX A
COST ESTIMATION



Face recognition on AWS

Estimate summary Info

Upfront cost	Monthly cost	Total 12 months cost
180.00 USD	31.91 USD	562.92 USD Includes upfront cost

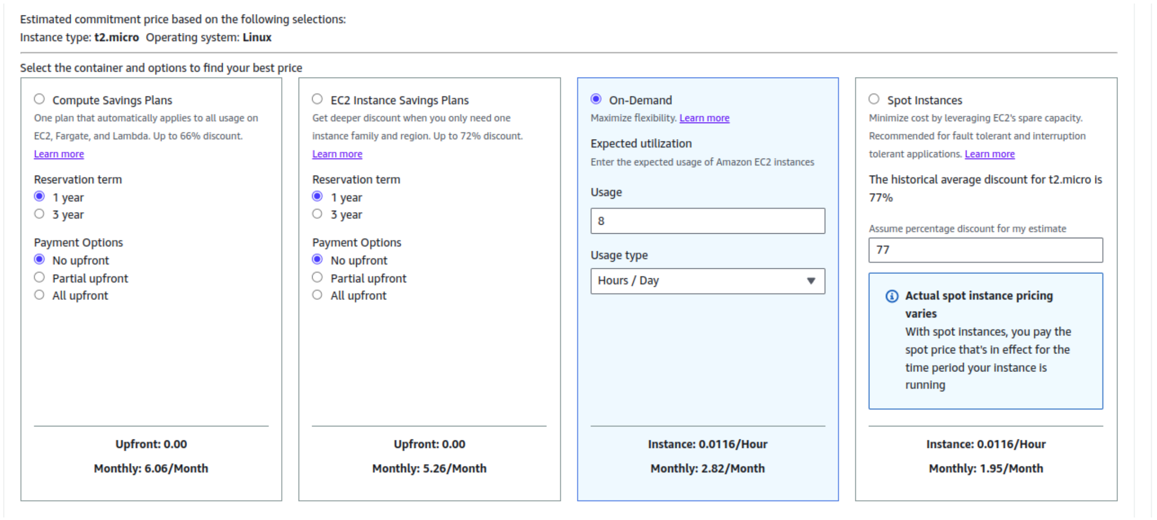
Getting Started with AWS

Face recognition on AWS

Service Name	Status	Upfront cost	Monthly cost	Description	Region	Config Summary
Amazon EC2	-	0.00 USD	4.23 USD	-	US East (Ohio)	Tenancy (Shared Instances), Operat…
Amazon Simple Storage Se…	-	0.00 USD	0.07 USD	-	US East (Ohio)	S3 Standard storage (3 GB per mon…
Amazon DynamoDB	-	180.00 USD	26.39 USD	-	US East (Ohio)	Table class (Standard), Average ite…
Amazon Rekognition	-	0.00 USD	1.20 USD	-	US East (Ohio)	Number of IndexFaces API calls per…
AWS Lambda	-	0.00 USD	0.00 USD	-	US East (Ohio)	Architecture (x86), Architecture (x8…
Amazon Simple Storage Se…	-	0.00 USD	0.02 USD	-	US East (Ohio)	S3 Standard storage (1 GB per mon…

Note. All prices are calculated without considering the free tier eligibility

A.1 EC2 t2.micro instance



Estimated commitment price based on the following selections:
Instance type: **t2.micro** Operating system: **Linux**

Select the container and options to find your best price

Compute Savings Plans
One plan that automatically applies to all usage on EC2, Fargate, and Lambda. Up to 66% discount. Learn more
Reservation term: [✓ 1 year] 3 year
Payment Options: [✓ No upfront] Partial upfront All upfront
Upfront: 0.00
Monthly: 6.06/Month

EC2 Instance Savings Plans
Get deeper discount when you only need one instance family and region. Up to 72% discount. Learn more
Reservation term: [✓ 1 year] 3 year
Payment Options: [✓ No upfront] Partial upfront All upfront
Upfront: 0.00
Monthly: 5.26/Month

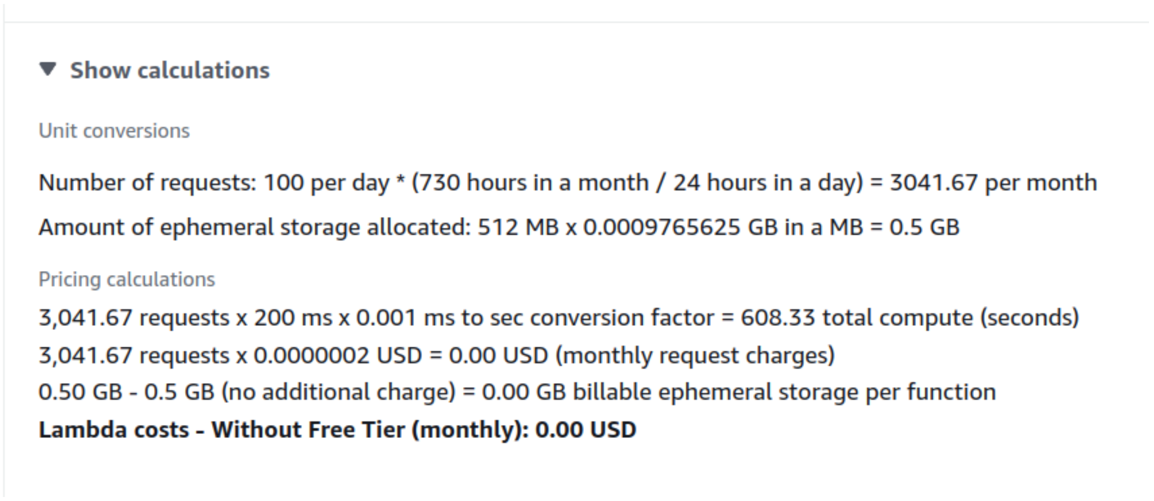
[✓ On-Demand]
Maximize flexibility. Learn more
Expected utilization
Enter the expected usage of Amazon EC2 instances
Usage: 8
Usage type: Hours / Day
Instance: 0.0116/Hour
Monthly: 2.82/Month

Spot Instances
Minimize cost by leveraging EC2's spare capacity. Recommended for fault tolerant and interruption tolerant applications. Learn more
The historical average discount for t2.micro is 77%
Assume percentage discount for my estimate: 77
Actual spot instance pricing varies
With spot instances, you pay the spot price that's in effect for the time period your instance is running
Instance: 0.0116/Hour
Monthly: 1.95/Month

- Chose t2.micro instance as high performance is not needed.
- Cost per day: $0.0116/hour * 1hour/day$ =0.0116/day
- Monthly cost: $0.0116/day * 30days/month$ =0.348/month

A.2 AWS lambda function

Lambda functions are priced based on the number of requests and the duration of execution.



▼ Show calculations

Unit conversions

Number of requests: 100 per day * (730 hours in a month / 24 hours in a day) = 3041.67 per month
Amount of ephemeral storage allocated: 512 MB x 0.0009765625 GB in a MB = 0.5 GB

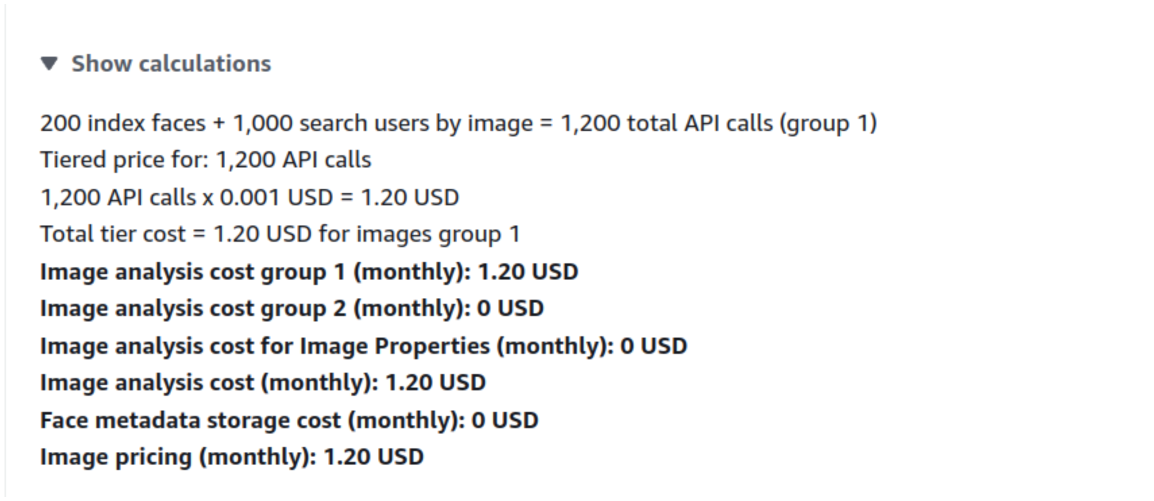
Pricing calculations

3,041.67 requests x 200 ms x 0.001 ms to sec conversion factor = 608.33 total compute (seconds)
3,041.67 requests x 0.0000002 USD = 0.00 USD (monthly request charges)
0.50 GB - 0.5 GB (no additional charge) = 0.00 GB billable ephemeral storage per function
Lambda costs - Without Free Tier (monthly): 0.00 USD

- Number of requests / day = 100
- Time of exceution = 200 ms
- Total amount = 0.000 USD

A.3 AWS rekognition

For the IndexFaces API, it charged based on the number of images processed and SearchFaces-ByImage API,irged based on the number of images compared.



▼ Show calculations

200 index faces + 1,000 search users by image = 1,200 total API calls (group 1)
Tiered price for: 1,200 API calls
1,200 API calls x 0.001 USD = 1.20 USD
Total tier cost = 1.20 USD for images group 1
Image analysis cost group 1 (monthly): 1.20 USD
Image analysis cost group 2 (monthly): 0 USD
Image analysis cost for Image Properties (monthly): 0 USD
Image analysis cost (monthly): 1.20 USD
Face metadata storage cost (monthly): 0 USD
Image pricing (monthly): 1.20 USD

- Number of calling IndexFaces API = 200
- Number of calling SearchFacesByImage = 1000
- Total amount = 1.2 USD

A.4 DynamoDB

DynamoDB pricing can be complex due to its various factors and pricing options.

- Choose mainly focus on the storage size for storing RekognitionIDs and Names.
- Storage 1 GB

A.5 S3 bucket

S3 is only used for storing images for registration purposes.

- Tiered price for: 1 GB ==> 1 GB x 0.023 USD = 0.02 USD

APPENDICIES A
DEMO WORK

This section demonstrates the practical implementation of the facial recognition system on AWS, showcasing the key components and their integration.

A.1 Infrastructure as Code (IaC) with AWS CloudFormation

To automate the deployment and management of AWS resources, a CloudFormation template is created in YAML format. The template defines the following resources:

- Amazon S3:
  - Create an S3 bucket to store uploaded images
  - Define bucket policies and lifecycle rules for cost optimization and data management
- AWS Lambda:
  - Create Lambda functions to trigger AWS Rekognition and interact with DynamoDB when images are uploaded to S3
- Amazon DynamoDB:
  - Create a DynamoDB table to store Rekognition IDs along with the corresponding names in a structured format
- Amazon Rekognition:
  - Create a Rekognition collection to support the IndexFaces and SearchFacesByImage APIs for efficient face recognition tasks
- Amazon EC2:
  - Create an EC2 instance to host the web server for the user interface
  - Define security group rules to control inbound and outbound traffic
- AWS Key Management Service (KMS):
  - Enable encryption of images stored in S3 using AWS KMS for enhanced security

The CloudFormation template ensures consistent and repeatable provisioning of the required AWS resources.

A.2 Web Server Hosting and Deployment

The facial recognition system's user interface is hosted on an EC2 instance. The following steps are performed:

1. Allocate an Elastic IP address and associate it with the EC2 instance for a stable and persistent public IP.
2. Deploy the application code (app.py) to the EC2 instance.

3. Test the application to ensure proper functionality and connectivity with other AWS services.

The source code for the facial recognition system is available on GitHub: `https://github.com/KaungHtetCho-22/Face-recognition-on-AWS`

A.3 Testing and Results

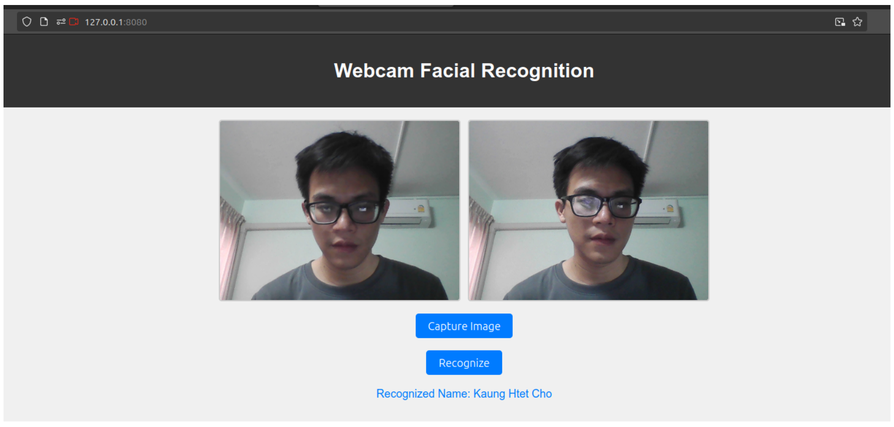
The facial recognition system is thoroughly tested to validate its functionality and performance. The following scenarios are covered:

1. User registration: Users can upload their facial images through the web interface, which are then processed by AWS Rekognition and stored in the S3 bucket.
2. Face recognition: When a user attempts to access a resource, their facial image is captured and compared against the registered faces using AWS Rekognition's SearchFacesByImage API. The system grants or denies access based on the recognition results.

A.4 Demo

The facial recognition system's web interface provides a seamless user experience for capturing images and performing face recognition. The demo showcases the following functionality:

- As soon as the webpage loads, the webcam is automatically opened, and the user is prompted to capture an image.
- The captured image is then used by the model to search for matching data in the AWS DynamoDB table, which stores the registered images and names.
- If the captured face is found in the database, the system displays a "recognized" message, indicating a successful match.
- The sample screenshot below shows the webpage running locally, demonstrating the functionality of the facial recognition system.



Recognized Name: Kaung Htet Cho

Figure A1 Sample screenshot of the facial recognition system's interface.