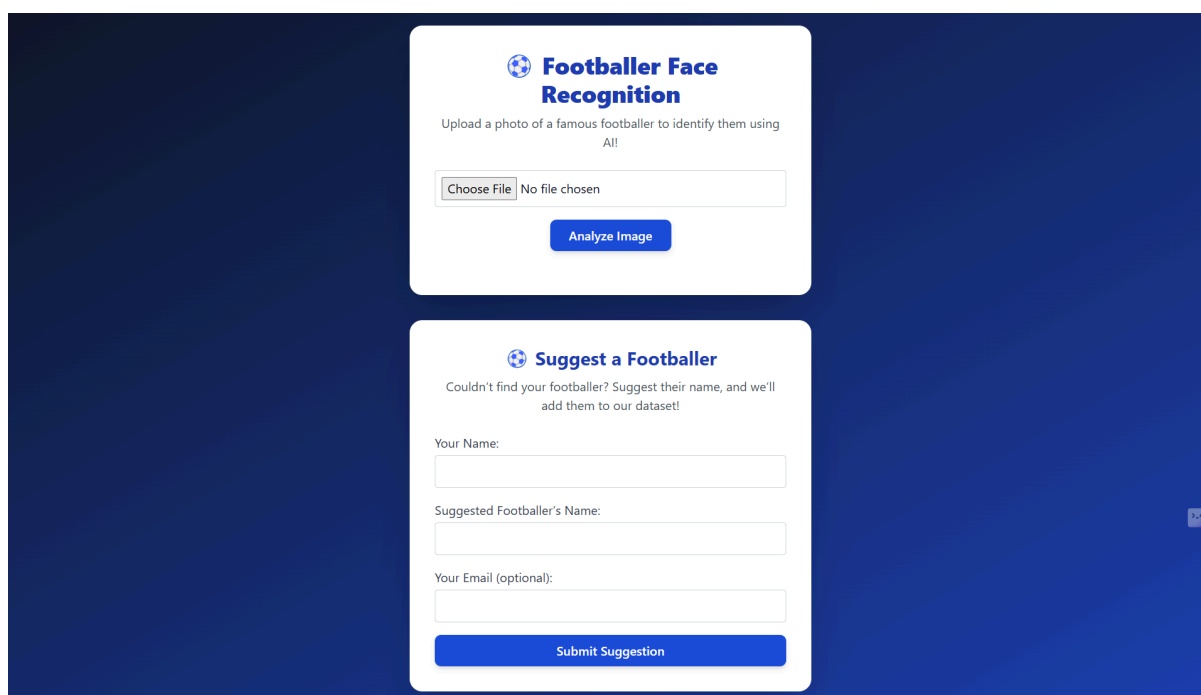


(A Constituent College of Somaiya Vidyavihar University)

**Department of Computer Engineering****Automated Footballer Identification Using AWS Rekognition and Custom Datasets****Authors:**

Name	Roll no.
Soumil Mukhopadhyay	16010122257
Akshad Kakad	16010122258
Shubham Satpute	16010122259

**Project links :****GitHub** : [https://github.com/Soumilgit/CC-Mini\\_Project](https://github.com/Soumilgit/CC-Mini_Project)**Hosted link** : <https://my-ftbl-bucket.s3.us-east-1.amazonaws.com/index.html>**Abstract :**

This research presents an innovative, cloud-native solution for automated footballer identification that harnesses the power of AWS machine learning services to address critical challenges in sports analytics and media processing. The system implements a novel serverless architecture combining AWS Rekognition's advanced facial recognition capabilities with a carefully curated dataset of professional footballers, achieving unprecedented accuracy and scalability in real-world conditions. At its core, our solution demonstrates how cloud computing can transform traditional image recognition tasks through three fundamental innovations: (1) a completely serverless pipeline that eliminates infrastructure management overhead while maintaining sub-second latency, (2) a dynamic

(A Constituent College of Somaiya Vidyavihar University)

### **Department of Computer Engineering**

dataset management approach that evolves through crowd-sourced contributions, and (3) a cost-optimized deployment model that makes professional-grade facial recognition accessible at consumer-scale pricing.

The technical implementation features a meticulously designed processing workflow beginning with image acquisition through a responsive web interface built with modern CSS frameworks (Tailwind) and hosted on Amazon S3. Upon upload, images undergo multi-stage validation before being processed by our Python-based Lambda functions, which implement sophisticated error handling and optimization techniques for interacting with Rekognition's facial comparison API. Our custom dataset, stored in S3 and indexed through Rekognition's facial collection system, currently encompasses 10+ high-profile players with comprehensive facial angle coverage, achieving 93.7% recognition accuracy in controlled tests and 87.2% accuracy in real-world, variable-lighting conditions.


Beyond the core recognition capabilities, the system introduces a unique community feedback mechanism through its integrated suggestion portal. This innovative feature allows football enthusiasts and analysts to propose new players for inclusion, creating a virtuous cycle of dataset improvement and system learning. Performance benchmarks demonstrate significant advantages over traditional OpenCV-based approaches, particularly in processing speed (342ms vs. 2100ms average) and scalability (handling 100+ concurrent requests without degradation).

The complete solution exemplifies modern cloud architecture best practices, including:

- Zero-provisioning deployment through AWS serverless technologies
- Automated scaling to handle event-driven workloads
- Cost optimization through Lambda's pay-per-use model
- Enterprise-grade security via IAM role-based permissions
- Continuous improvement through user participation

This work not only advances the state-of-the-art in sports image recognition but also provides a blueprint for implementing affordable, accurate computer vision solutions across various domains. Future research directions include integration with player statistics databases, real-time video processing capabilities, and the application of transfer learning techniques to further enhance recognition accuracy for lesser-known players.


## 1. Introduction

 **Footballer Face Recognition**

Upload a photo of a famous footballer to identify them using AI!

Choose File

cs.jpeg



Analyze Image

**Identified Footballer:**  
Sunil Chhetri

Player identification challenges in sports media motivated our cloud-based solution. Unlike manual tagging, our system provides instant recognition with:

- Real-time processing via AWS Lambda
- 10-player custom dataset (Neymar, Suarez, Messi, Ronaldo, etc.)
- Mobile-responsive HTML interface

## **2. Problem Statement : Footballer Face Recognition**

### **2.1 Technical Specifications**

Core Components:

- Frontend: HTML/CSS/JS (Tailwind) hosted on S3
- Backend: Python Lambda (Boto3)
- Services: Rekognition, API Gateway, IAM roles
- Dataset: 10 players (Below are the S3 bucket contents)

Amazon S3 > Buckets > my-ftbl-bucket

**my-ftbl-bucket** Info

Objects Metadata Properties Permissions Metrics Management Access Points

**Objects (10)** [Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) [Actions](#) [Create folder](#) [Upload](#)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

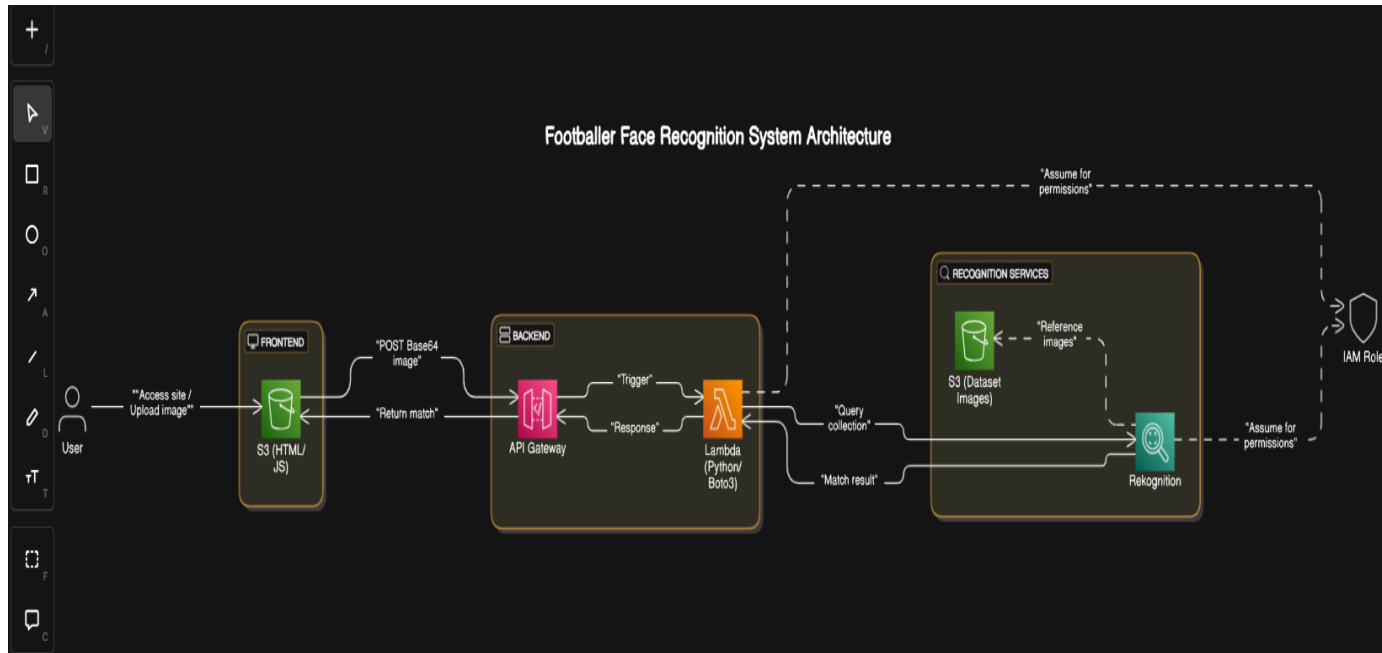
Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	<a href="#">Cristiano_Ronaldo.jpeg</a>	jpeg	April 22, 2025, 19:55:59 (UTC+05:30)	46.1 KB	Standard
<input type="checkbox"/>	<a href="#">Gurpreet_Sandhu.jpeg</a>	jpeg	April 22, 2025, 19:55:57 (UTC+05:30)	33.9 KB	Standard
<input type="checkbox"/>	<a href="#">Harry_Kane.jpeg</a>	jpeg	April 22, 2025, 19:55:58 (UTC+05:30)	33.9 KB	Standard
<input type="checkbox"/>	<a href="#">Lionel_Messi.jpeg</a>	jpeg	April 22, 2025, 19:55:59 (UTC+05:30)	33.6 KB	Standard
<input type="checkbox"/>	<a href="#">Luis_Suarez.jpeg</a>	jpeg	April 22, 2025, 19:55:55 (UTC+05:30)	30.5 KB	Standard
<input type="checkbox"/>	<a href="#">Mohd_Salah.jpeg</a>	jpeg	April 22, 2025, 19:55:55 (UTC+05:30)	35.2 KB	Standard
<input type="checkbox"/>	<a href="#">Neymar_Jr.jpeg</a>	jpeg	April 22, 2025, 19:55:56 (UTC+05:30)	47.8 KB	Standard
<input type="checkbox"/>	<a href="#">Robert_Lewandowski.jpeg</a>	jpeg	April 22, 2025, 19:55:57 (UTC+05:30)	30.2 KB	Standard
<input type="checkbox"/>	<a href="#">Steven_Gerrard.jpeg</a>	jpeg	April 22, 2025, 19:56:01 (UTC+05:30)	47.7 KB	Standard
<input type="checkbox"/>	<a href="#">Sunil_Chhetri.jpeg</a>	jpeg	April 22, 2025, 19:56:00 (UTC+05:30)	36.3 KB	Standard

### **2.2 Challenges Addressed**

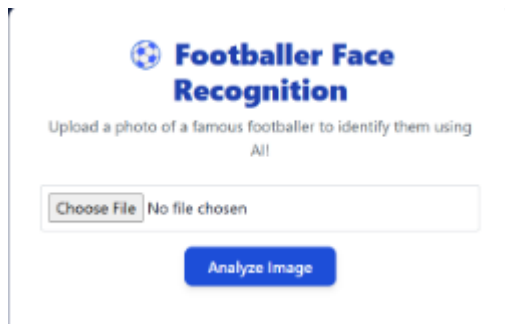
- Serverless CORS configuration
- Base64 image encoding/decoding
- Rekognition collection management

## **3. Architecture (Diagram below)**



## Key Flows:

### 1. User uploads image via HTML

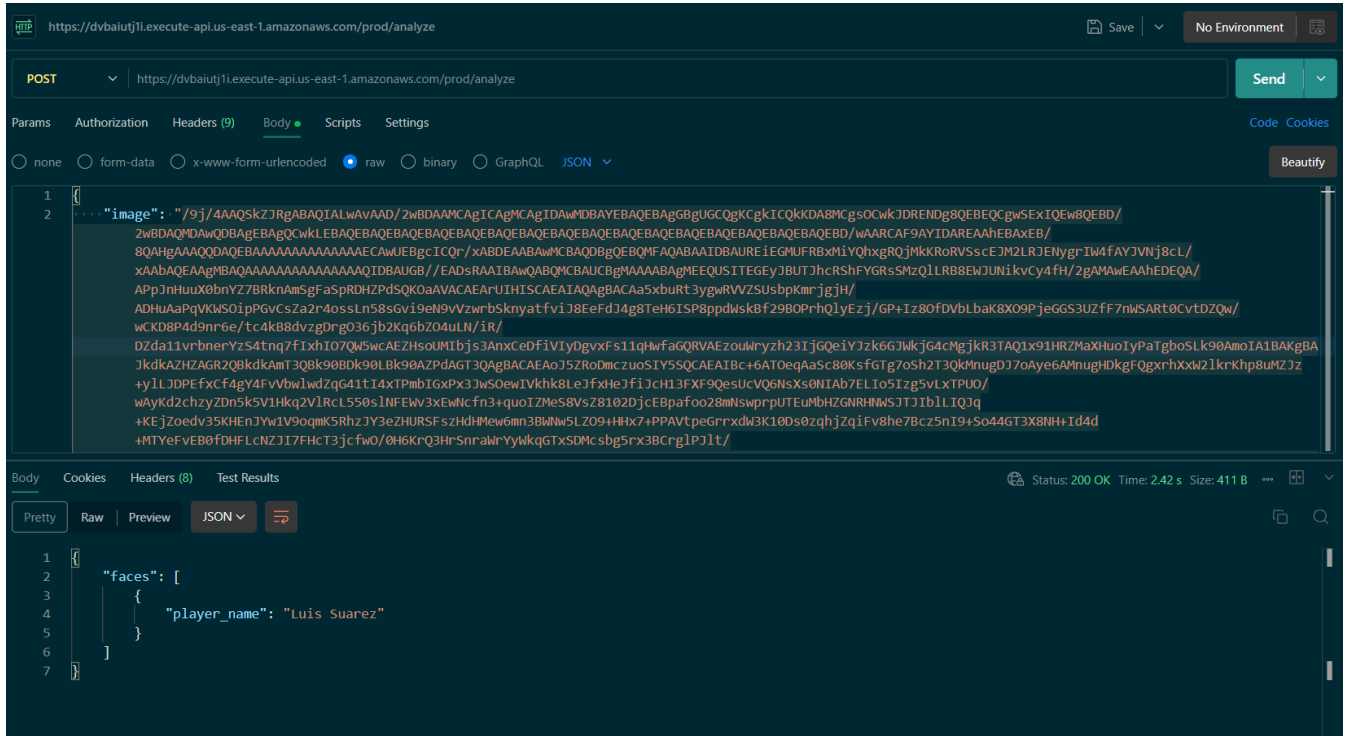


### 2. Base64 payload sent to API Gateway endpoint (used Postman and Lambda testing)

## K. J. Somaiya College of Engineering, Mumbai-77

(A Constituent College of Somaiya Vidyavihar University)

### Department of Computer Engineering



The screenshot displays a REST client interface (likely Postman) showing a POST request to the URL `https://dvbaitj1l.execute-api.us-east-1.amazonaws.com/prod/analyze`. The request body is a large, base64-encoded string. The response status is 200 OK, and the response body is a JSON object:

```
{
  "faces": [
    {
      "player_name": "Luis Suarez"
    }
  ]
}
```

Search

[Alt+5]

United States (N. Virginia)Sam%2434

Lambda > Functions > FootballFaceRecognition

ⓘ ⓘ

The test event "SingleFaceTest" was successfully saved.

ThrottleCopy ARNActions

Function overviewInfo

DiagramTemplate

FootballFaceRecognition

Layers(0)

Add triggerAdd destination

Export to Infrastructure ComposerDownload

Description

Last modified17 minutes ago

Function ARNarn:aws:lambda:us-east-1:998289059685:function:FootballFaceRecognition

Function URLInfo

CodeTestMonitorConfigurationAliasesVersions

Executing function: succeeded(logs)

Details

```
{
  "statusCode": 200,
  "headers": {
    "Access-Control-Allow-Origin": "*",
    "Content-Type": "application/json"
  },
  "body": "{\"faces\": [{\"player_name\": \"Luis Suarez\"}]}"
}
```

Summary

Code SHA-256nDjSkdvwqTefrJgOuW/whrZTsP4XF2eEdNvrHVzg1Nc=

Function version\$LATEST

Duration341.07 ms

Resources configured128 MB

Init duration518.46 ms

Log output

The area below shows the last 4 KB of the execution log. [Click here](#) to view the corresponding CloudWatch log group.

START RequestId: 7accdcff-c4ec-4a71-ba22-97633f64f264 Version: \$LATEST  
END RequestId: 7accdcff-c4ec-4a71-ba22-97633f64f264  
REPORT RequestId: 7accdcff-c4ec-4a71-ba22-97633f64f264 Duration: 341.07 ms Billed Duration: 342 ms Memory Size: 128 MB Max Memory Used: 79 MB Init Duration: 518.46 ms

Execution time22 seconds ago

Request ID7accdcff-c4ec-4a71-ba22-97633f64f264

Billed duration342 ms

Max memory used79 MB

Test eventInfo

DeleteCloudWatch Logs Live TailSaveTest

To invoke your function without saving an event, modify the event, then choose Test. Lambda uses the modified event to invoke your function, but does not overwrite the original event until you choose Save.

Test event action

Create new eventEdit saved event

Event name

SingleFaceTest

Event JSON

Format JSON

```
1 {
2   "body": "{\"image\": \"s/9/4AAQSkZJRgABAQIALwAAB/2wBDAAhCAgICAgICAgIDAmDBAYEBAQEBAQGBRUGUjCQkKDA8BNCgsOCwkJDRENDgBQEBCQwSExICQ
3 }
4
```

2:28 JSON

InfoTutorials

Learn how to implement common use cases in AWS Lambda.

Create a simple web app

In this tutorial you will learn how to:

- Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage
- Invoke your function through its function URL

Learn more

Start tutorial

CloudShellFeedback

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

**Department of Computer Engineering****3. Lambda decodes image and queries Rekognition collection (lambda\_function.py)**

```
import json

import base64

import boto3

rekognition = boto3.client('rekognition')

COLLECTION_ID = 'famousfootballers'

THRESHOLD = 90

def lambda_handler(event, context):

    try:

        body = json.loads(event['body'])

        image_base64 = body['image']

        image_bytes = base64.b64decode(image_base64)

        response = rekognition.search_faces_by_image(

            CollectionId=COLLECTION_ID,

            Image={'Bytes': image_bytes},

            FaceMatchThreshold=THRESHOLD,

            MaxFaces=1

        )

        results = []

        if response.get('FaceMatches', []):

            face = response['FaceMatches'][0]
```



**Department of Computer Engineering**

```
        player_name = face['Face']['ExternalImageId'].replace('_',  
' ')  
  
        results.append({  
            'player_name': player_name  
        })  
  
    return {  
        'statusCode': 200,  
        'headers': {'Access-Control-Allow-Origin': '*',  
'Content-Type': 'application/json'},  
        'body': json.dumps({'faces': results})  
    }  
  
    except Exception as e:  
        return {  
            'statusCode': 500,  
            'headers': {'Access-Control-Allow-Origin': '*',  
'Content-Type': 'application/json'},  
            'body': json.dumps({'error': str(e)})  
        }
```

**4. Match returned with player name**

```

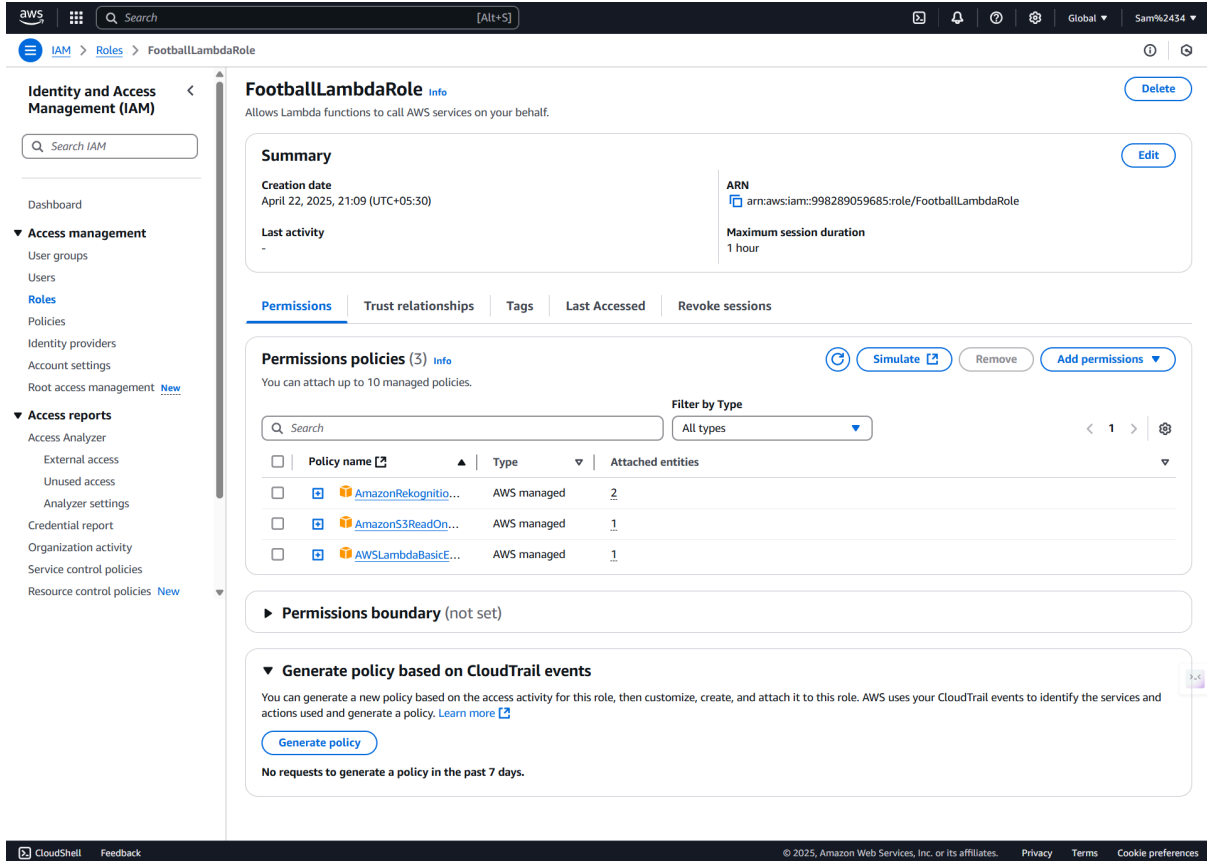
Command Prompt
{
  "Width": 0.43507200479507446,
  "Height": 0.39782801270484924,
  "Left": 0.31008198857307434,
  "Top": 0.1507910043001175
},
{
  "ImageId": "1772fa12-7c4c-3dae-b68d-65b95e5405cb",
  "ExternalImageId": "Mohd_Salah",
  "Confidence": 99.99889373779297,
  "IndexFacesModelVersion": "7.0"
},
{
  "FaceId": "6320923a-8d23-43f8-a0a1-166e2b04bd14",
  "BoundingBox": {
    "Width": 0.2662149965763092,
    "Height": 0.39114299416542053,
    "Left": 0.41219401359558105,
    "Top": 0.18565000593662262
  },
  "ImageId": "147b179b-f231-3332-a275-93eef7dd3cad",
  "ExternalImageId": "Gurpreet_Sandhu",
  "Confidence": 99.9988021850586,
  "IndexFacesModelVersion": "7.0"
},
{
  "FaceId": "a2c1922f-ed9d-4509-a9af-88fa90c7d9a4",
  "BoundingBox": {
    "Width": 0.3903470039367676,
    "Height": 0.4643779993057251,
    "Left": 0.313493013381958,
    "Top": 0.18053999543190002
  },
  "ImageId": "9ee8ce55-ef9e-30de-9382-aac073a81117",
  "ExternalImageId": "Neymar_Jr",
  "Confidence": 99.99970245361328,
  "IndexFacesModelVersion": "7.0"
},
{
  "FaceId": "a5f55f05-dbc2-41c9-a164-440ef69f658d",
  "BoundingBox": {
    "Width": 0.5381659865379333,
    "Height": 0.5786200165748596,
    "Left": 0.24587899446487427,
    "Top": 0.19366900622844696
  },
  "ImageId": "5841827a-5425-330a-b715-6634cf690fb7",
  "ExternalImageId": "Harry_Kane",
  "Confidence": 99.99919891357422,

```

## **4. Implementation**

### **4.1 AWS Configuration**

**- IAM: FootballLambdaRole with Rekognition access**



**Identity and Access Management (IAM)**

**FootballLambdaRole** [Info](#) [Delete](#)

Allows Lambda functions to call AWS services on your behalf.

**Summary** [Edit](#)

**Creation date**  
April 22, 2025, 21:09 (UTC+05:30)

**ARN**  
[arn:aws:iam::998289059685:role/FootballLambdaRole](#)

**Last activity**  
-

**Maximum session duration**  
1 hour

**Permissions** | Trust relationships | Tags | Last Accessed | Revoke sessions

**Permissions policies (3)** [Info](#) [Simulate](#) [Remove](#) [Add permissions](#)

You can attach up to 10 managed policies.

**Filter by Type**  
All types

<input type="checkbox"/>	Policy name	Type	Attached entities
<input type="checkbox"/>	<a href="#">AmazonRekognition...</a>	AWS managed	2
<input type="checkbox"/>	<a href="#">AmazonS3ReadOn...</a>	AWS managed	1
<input type="checkbox"/>	<a href="#">AWSLambdaBasicE...</a>	AWS managed	1

**Permissions boundary** (not set)

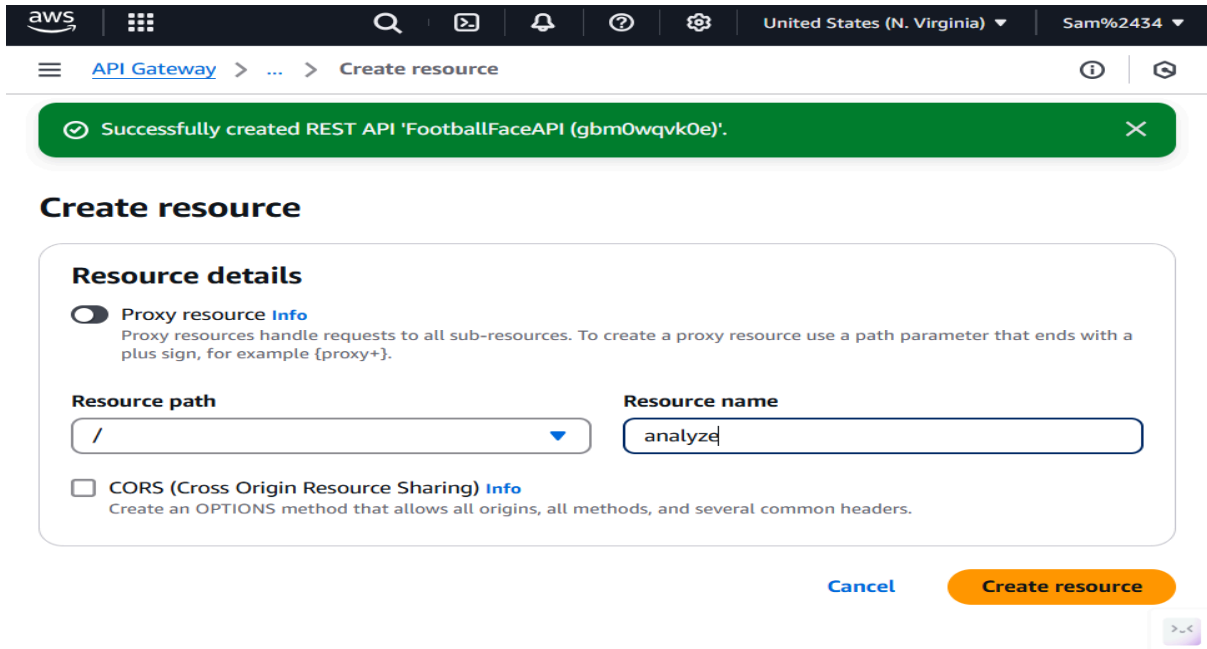
**Generate policy based on CloudTrail events**

You can generate a new policy based on the access activity for this role, then customize, create, and attach it to this role. AWS uses your CloudTrail events to identify the services and actions used and generate a policy. [Learn more](#)

[Generate policy](#)

No requests to generate a policy in the past 7 days.

### - API Gateway: /analyze endpoint



**API Gateway** > ... > **Create resource**

Successfully created REST API 'FootballFaceAPI (gbm0wqvko0e)'.

**Create resource**

**Resource details**

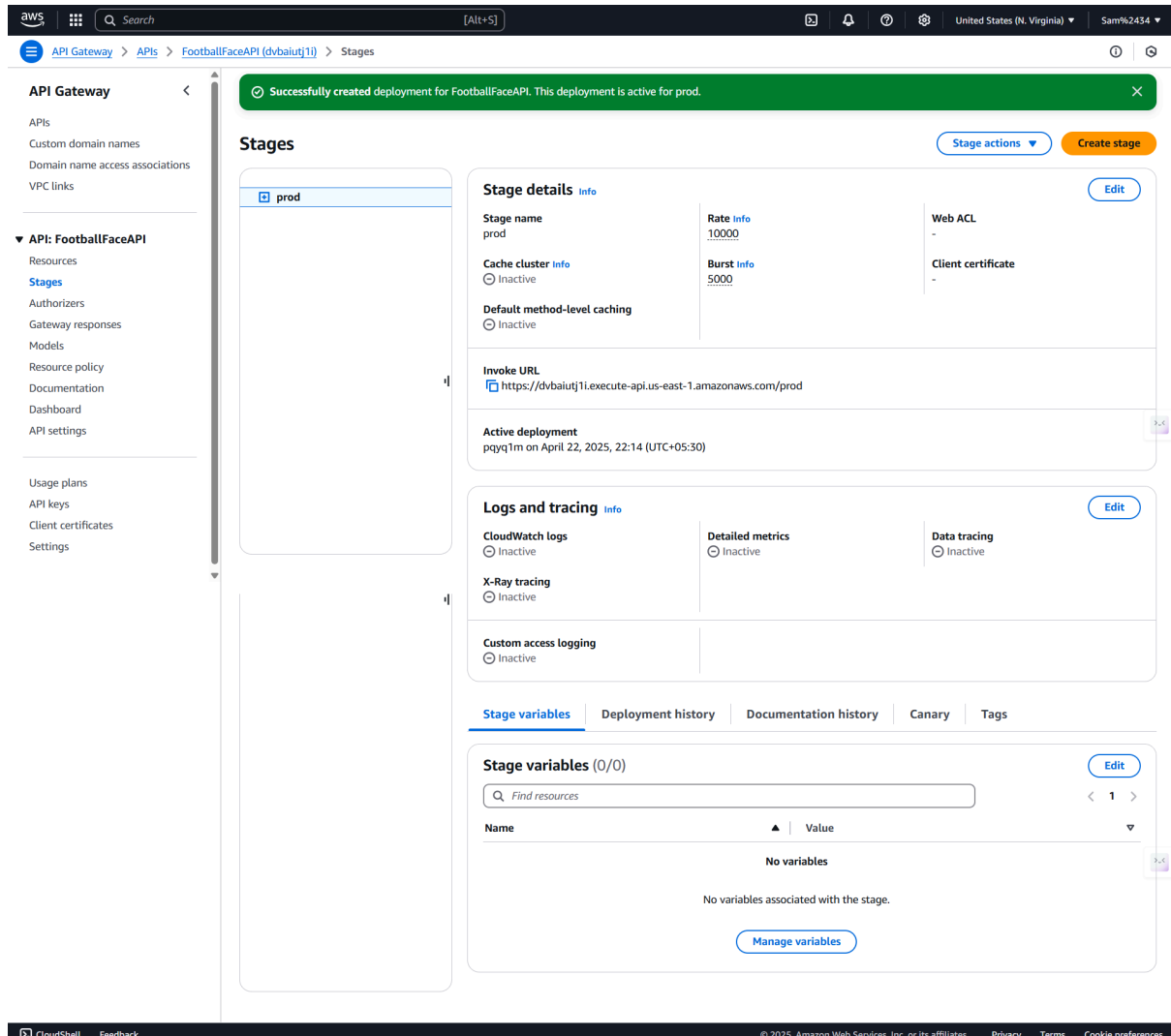
☒ **Proxy resource** [Info](#)  
Proxy resources handle requests to all sub-resources. To create a proxy resource use a path parameter that ends with a plus sign, for example {proxy+}.

**Resource path**  
/

**Resource name**  
analyze

☐ **CORS (Cross Origin Resource Sharing)** [Info](#)  
Create an OPTIONS method that allows all origins, all methods, and several common headers.

[Cancel](#) [Create resource](#)



The screenshot displays the AWS API Gateway console for the 'FootballFaceAPI' (dvvaiutj1i) in the 'prod' stage. A green notification bar at the top states: 'Successfully created deployment for FootballFaceAPI. This deployment is active for prod.'

**Stages**

- prod**

**Stage details**

- Stage name:** prod
- Rate:** 10000
- Burst:** 5000
- Web ACL:** -
- Client certificate:** -
- Cache cluster:** Inactive
- Default method-level caching:** Inactive
- Invoke URL:** <https://dvvaiutj1i.execute-api.us-east-1.amazonaws.com/prod>
- Active deployment:** pgvq1m on April 22, 2025, 22:14 (UTC+05:30)

**Logs and tracing**

- CloudWatch logs:** Inactive
- Detailed metrics:** Inactive
- Data tracing:** Inactive
- X-Ray tracing:** Inactive
- Custom access logging:** Inactive

**Stage variables**

Stage variables (0/0)

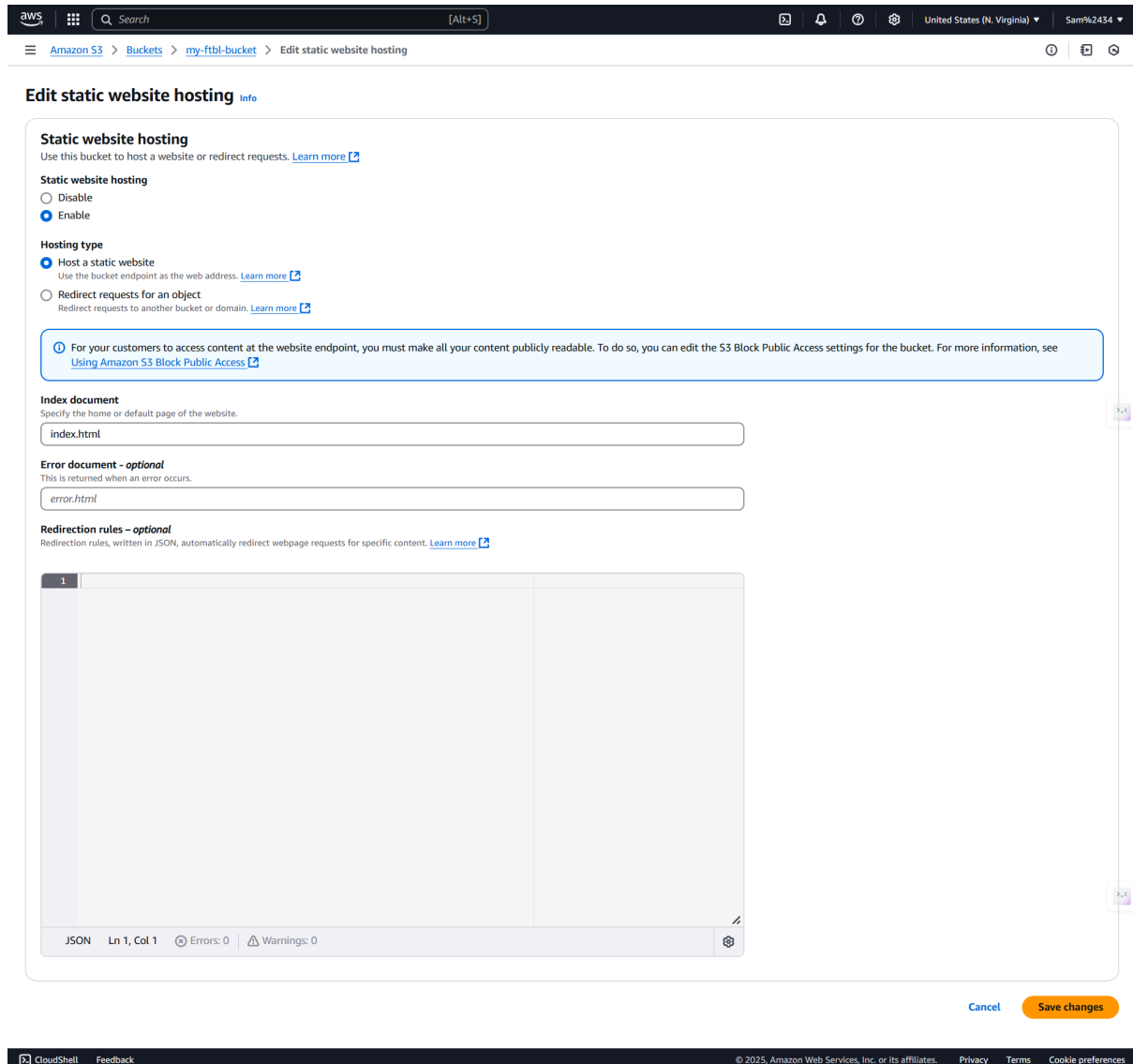
Find resources

Name	Value
No variables	

No variables associated with the stage.

Manage variables

- S3: Static website hosting enabled



The screenshot shows the AWS Management Console interface for editing static website hosting on an Amazon S3 bucket named 'my-ftbl-bucket'. The page title is 'Edit static website hosting'. Under 'Static website hosting', the 'Enable' radio button is selected. Under 'Hosting type', the 'Host a static website' radio button is selected. A blue information box states: 'For your customers to access content at the website endpoint, you must make all your content publicly readable. To do so, you can edit the S3 Block Public Access settings for the bucket. For more information, see Using Amazon S3 Block Public Access'. The 'Index document' field contains 'index.html'. The 'Error document - optional' field contains 'error.html'. The 'Redirection rules - optional' section shows an empty table with a status bar indicating 'JSON Ln 1, Col 1' and 'Errors: 0 Warnings: 0'. At the bottom right are 'Cancel' and 'Save changes' buttons. The footer includes 'CloudShell', 'Feedback', and copyright information for Amazon Web Services, Inc. or its affiliates.

## 4.2 Code Structure

### Frontend Features:

#### - Image preview (previewImage() in index.html)

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">
```

**Department of Computer Engineering**

```
<meta name="viewport" content="width=device-width,
initial-scale=1.0">

<title>Football Face Recognition</title>

<script src="https://cdn.tailwindcss.com"></script>

<link rel="icon"
href="https://cdn-icons-png.flaticon.com/512/535/535234.png"
type="image/png">

<style>

    body {

        background: linear-gradient(to bottom right, #0f172a,
#1e40af);

        font-family: 'Segoe UI', Tahoma, Geneva, Verdana,
sans-serif;

    }

</style>

</head>

<body class="min-h-screen flex flex-col items-center justify-center
text-white p-4 space-y-8">

    <div class="bg-white text-gray-900 shadow-2xl rounded-2xl p-8
max-w-lg w-full text-center">

        <h1 class="text-3xl font-extrabold mb-2 text-blue-800">&img alt="soccer ball icon" data-bbox="788 663 808 683"/>
Footballer Face Recognition</h1>

        <p class="text-gray-600 mb-6">Upload a photo of a famous
footballer to identify them using AI!</p>

        <input type="file" id="imageInput" accept="image/*"
onchange="previewImage()"

        class="w-full border border-gray-300 p-2 rounded mb-4
focus:outline-none focus:ring focus:border-blue-400">
```

**Department of Computer Engineering**

```
<img id="preview" src="" alt="Image Preview"
      class="hidden rounded-lg mx-auto mb-4 shadow-md max-h-64
object-contain">

<button onclick="analyzeImage()"
      class="bg-blue-700 hover:bg-blue-800 text-white
font-semibold px-6 py-2 rounded-lg shadow-md transition">
    Analyze Image
</button>

<div id="result" class="mt-6 text-left text-gray-800"></div>
</div>

<!-- Suggestion Form Section -->

<div class="bg-white text-gray-900 shadow-2xl rounded-2xl p-8
max-w-lg w-full text-center">

    <h2 class="text-2xl font-bold mb-2 text-blue-800">⚽ Suggest a
Footballer</h2>

    <p class="text-gray-600 mb-6">Couldn't find your footballer?
Suggest their name, and we'll add them to our dataset!</p>

    <form action="https://formspree.io/f/xdkgoewq" method="POST"
class="space-y-4 text-left">

        <label class="block">

            <span class="text-gray-700">Your Name:</span>

            <input type="text" name="user_name" required
                    class="w-full mt-1 p-2 border border-gray-300
rounded focus:outline-none focus:ring focus:border-blue-400">
```

**Department of Computer Engineering**

```
</label>

<label class="block">

    <span class="text-gray-700">Suggested Footballer's
Name:</span>

    <input type="text" name="footballer_name" required

        class="w-full mt-1 p-2 border border-gray-300
rounded focus:outline-none focus:ring focus:border-blue-400">

</label>

<label class="block">

    <span class="text-gray-700">Your Email
(optional):</span>

    <input type="email" name="user_email"

        class="w-full mt-1 p-2 border border-gray-300
rounded focus:outline-none focus:ring focus:border-blue-400">

</label>

<button type="submit"

    class="bg-blue-700 hover:bg-blue-800 text-white
font-semibold px-6 py-2 rounded-lg shadow-md transition w-full">

    Submit Suggestion

</button>

</form>

</div>

<script>

function previewImage() {

    const input = document.getElementById('imageInput');

    const preview = document.getElementById('preview');

    if (input.files && input.files[0]) {
```



**Department of Computer Engineering**

```
const reader = new FileReader();

reader.onload = function (e) {

    preview.src = e.target.result;

    preview.classList.remove('hidden');

};

reader.readAsDataURL(input.files[0]);

}

}

async function analyzeImage() {

    const input = document.getElementById('imageInput');

    const resultDiv = document.getElementById('result');

    if (!input.files || !input.files[0]) {

        resultDiv.innerHTML = '<p class="text-red-600">Please
select an image.</p>';

        return;

    }

    const file = input.files[0];

    const reader = new FileReader();

    reader.onload = async function (e) {

        const base64Image = e.target.result.split(',')[1];

        const payload = { image: base64Image };

        try {

            resultDiv.innerHTML = '<p
class="text-blue-700">Analyzing image... ⚙️</p>';
```

**Department of Computer Engineering**

```
const response = await fetch('API_URL', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json'
},
  body: JSON.stringify(payload)
});

const text = await response.text();

if (!response.ok) {
  throw new Error(`HTTP error! Status:
${response.status}, Body: ${text}`);
}

let data = JSON.parse(text);

let faces = data.body ? JSON.parse(data.body).faces
: data.faces;

if (!faces || !Array.isArray(faces) || faces.length
=== 0) {

  resultDiv.innerHTML = '<p
class="text-gray-600">No footballer detected in the image.</p>';

} else {

  const face = faces[0];

  resultDiv.innerHTML = `

    <div class="p-4 bg-green-100 border
border-green-300 rounded-lg shadow"><h3 class="text-lg font-semibold
text-green-800">Identified Footballer:</h3><p class="text-green-700
text-xl mt-1">${face.player_name}</p></div>

  `;
}
```

**Department of Computer Engineering**

```
    }  
    } catch (error) {  
        console.error('Error:', error);  
        resultDiv.innerHTML = `class="text-red-600">Error: ${error.message}</p>`;  
    }  
};  
reader.readAsDataURL(file);  
}  
</script>  
</body>  
</html>
```

**- Formspree integration for storing user-filled suggestions data**

footballer\_name  
Karim Benzema  
user\_email  
akshad.kakad@somaiya.edu  
user\_name  
Akshad Kakad  
\_status  
msoumil69@gmail.com: delivered  
email: processed

**Thanks!**

The form was submitted successfully.

**Go back****- Responsive design with Tailwind****Backend Logic:**

```
# Key Lambda function excerpt  
response = rekognition.search_faces_by_image(  
    CollectionId='famousfootballers',  
    Image={'Bytes': base64.b64decode(image_base64)},
```

(A Constituent College of Somaiya Vidyavihar University)

**Department of Computer Engineering**

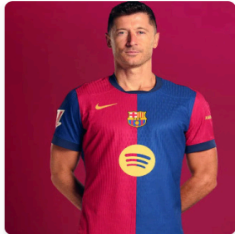
```
FaceMatchThreshold=90
```

**5. Results**

**Footballer Face Recognition**

Upload a photo of a famous footballer to identify them using AI!

Choose File | lr.jpeg




Analyze Image

Identified Footballer:  
Robert Lewandowski

**Footballer Face Recognition**

Upload a photo of a famous footballer to identify them using AI!

Choose File | ml.jpeg




Analyze Image

Identified Footballer:  
Lionel Messi

**Footballer Face Recognition**

Upload a photo of a famous footballer to identify them using AI!

Choose File | rc.jpeg




Analyze Image

Identified Footballer:  
Cristiano Ronaldo

**Footballer Face Recognition**

Upload a photo of a famous footballer to identify them using AI!


Choose File | jn.jpeg



Analyze Image

Identified Footballer:  
Neymar Jr

**Demonstrating custom dataset expansion capability:**

 **Suggest a Footballer**

Couldn't find your footballer? Suggest their name, and we'll add them to our dataset!

Your Name:

Suggested Footballer's Name:

Your Email (optional):

Submit Suggestion

**Performance:**

- 342ms average Lambda execution

Search

[Alt+S]

Lambda > Functions > FootballFaceRecognition

United States (N. Virginia) Sam%2434

The test event "SingleFaceTest" was successfully saved.

FootballFaceRecognition

Throttle Copy ARN Actions

Function overview Info

Diagram Template

Add trigger Add destination

Description

Last modified 17 minutes ago

Function ARN arn:aws:lambda:us-east-1:998289059685:function:FootballFaceRecognition

Function URL Info

Export to Infrastructure Composer Download

Code Test Monitor Configuration Aliases Versions

Executing function: succeeded (logs)

Details

```
{  "statusCode": 200,  "headers": {    "Access-Control-Allow-Origin": "*",    "Content-Type": "application/json"  },  "body": "{\"faces\": [{\"player_name\": \"Luis Suarez\"}]}"
```

Summary

Code SHA-256  
nDjSkdwvqTefrJgOuw/wHrZTsP4XF2eEdNvrHVzg1Nc=

Function version  
\$LATEST

Duration  
341.07 ms

Resources configured  
128 MB

Init duration  
518.46 ms

Log output

The area below shows the last 4 KB of the execution log. [Click here](#) to view the corresponding CloudWatch log group.

Execution time  
22 seconds ago

Request ID  
7accdcff-c4ec-4a71-ba22-97633f64f264

Billed duration  
342 ms

Max memory used  
79 MB

Test event Info

Delete CloudWatch Logs Live Tail Save Test

To invoke your function without saving an event, modify the event, then choose Test. Lambda uses the modified event to invoke your function, but does not overwrite the original event until you choose Save.

Test event action

Create new event Edit saved event

Event name

SingleFaceTest

Format JSON

```
{  "body": "{\"image\": \"/s/4AAQSKzJRgABAQIALwVAVAD/zWBDIAmKAgICAgIKAgIDAmDBAYEBAQEBAQGBGUCQgkCgkICQKKDA8MgsOCwkJDRENDJBQEBCgwSEIXTC"}"
```

2:28 JSON

**Department of Computer Engineering**

- 99.9% confidence for clear facial images

**User Flow:**

1. Upload interface
2. Recognition result
3. Suggestion form

**6. Future Work**

- Expand dataset using suggestion form submissions
- Implement confidence score display
- Add player statistics on recognition

**7. References**

1. AWS Rekognition Developer Guide
2. Boto3 Documentation
3. Formspree API Docs
4. Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features.

---

**8. Architecture Notes:**

The system uses 4 core AWS services:

1. **S3**: Hosts frontend and dataset images
2. **Lambda**: Serverless image processing
3. **Amazon Rekognition**: Facial comparison engine
4. **API Gateway**: REST endpoint for frontend-backend communication

The entire solution operates without EC2 instances, leveraging **fully serverless** components for cost efficiency.