PROJECT ENTITLED

Face Detection Attendance Management System

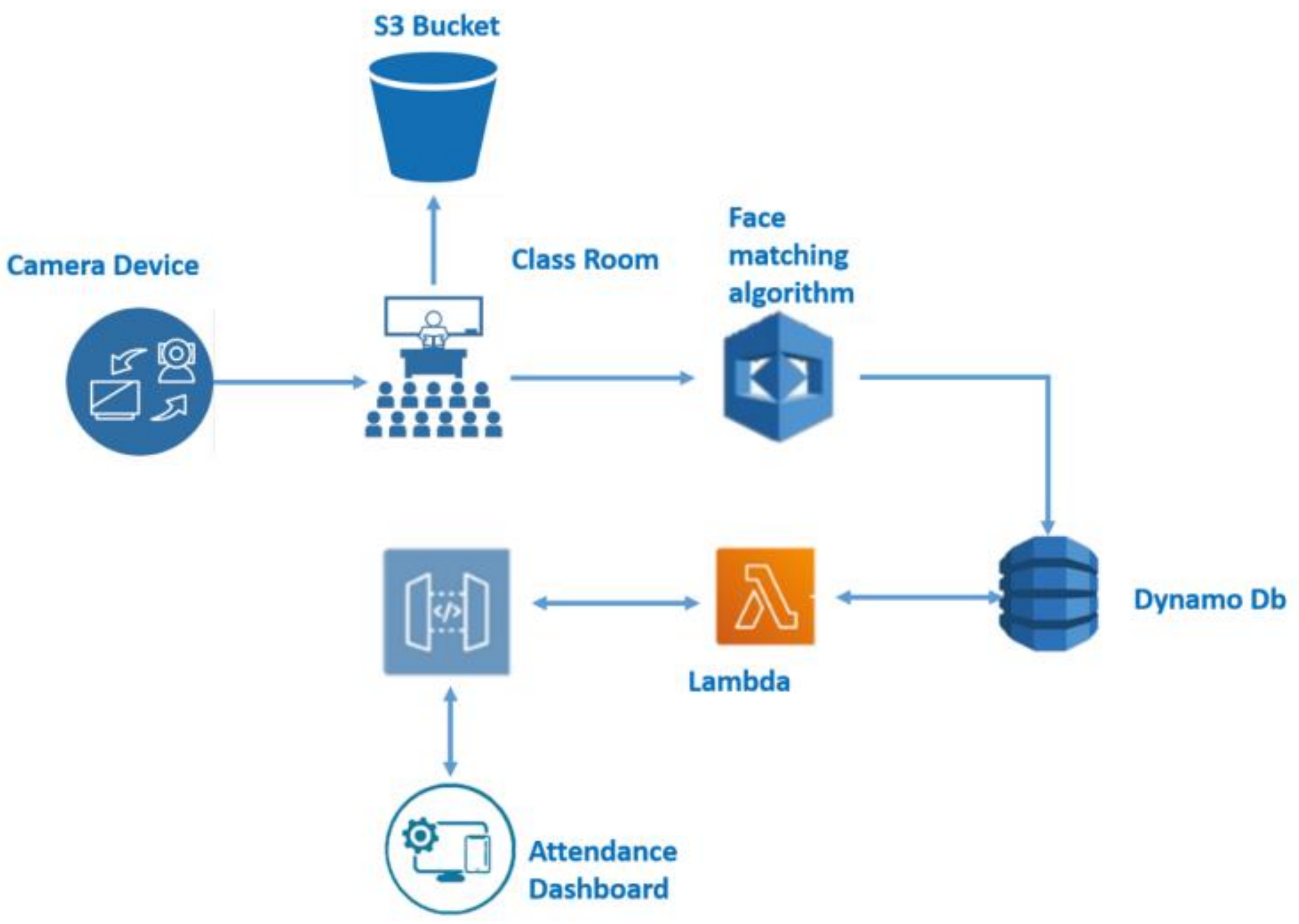
Group Members Name

1) Rajan Pal
2) Krishnakant Soni
3) Shweta Rokade
4) Urvashi
5) Ritesh Devre

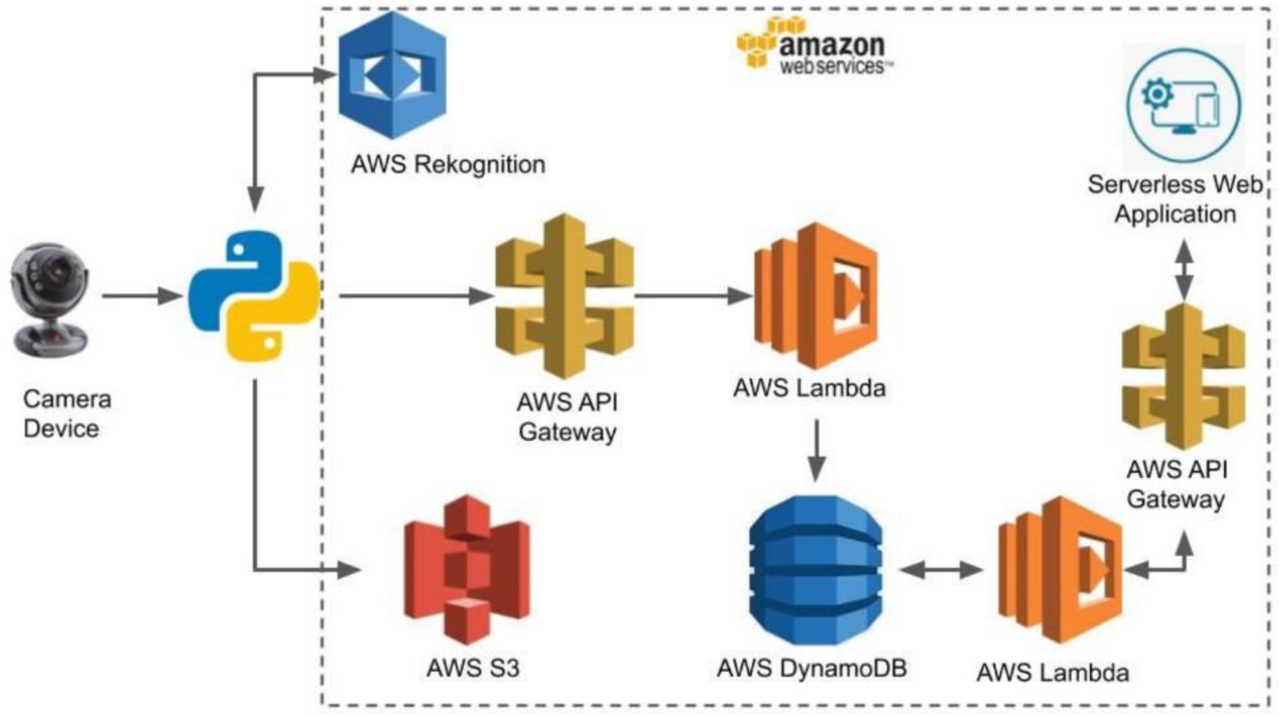
Introduction

The application shall capture hourly attendance without any manual intervention. develop a smart device that can be integrated with a camera that will capture the images of class for every hour and send the images to model. Then the model will use AWS Rekognition Service to recognize the student's faces & push the images to S3(Simple Storage Service) for storage and also updates the attendance automatically in a database. build a web-based dashboard to visualize all the student's attendance information.

Block Diagram :



Flowchart

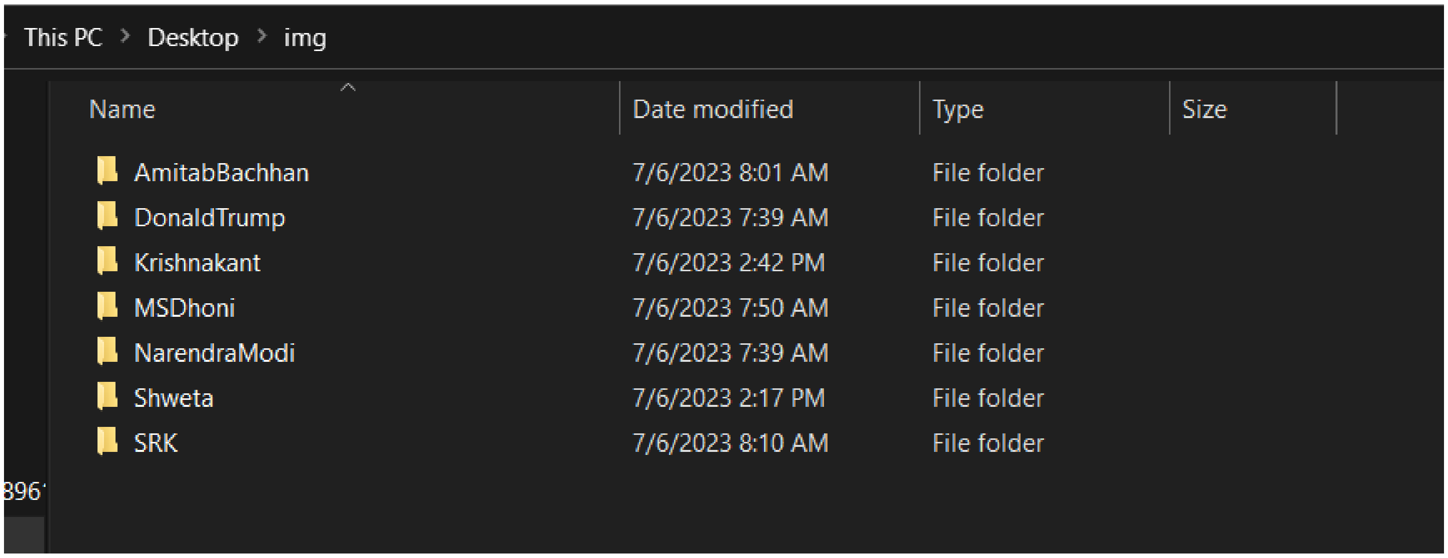

amazon web services
AWS Rekognition
Serverless Web Application
Camera Device
AWS API Gateway
AWS Lambda
AWS API Gateway
AWS S3
AWS DynamoDB
AWS Lambda

Implementation

Steps :

A] For Adding Data in Dynamodb :

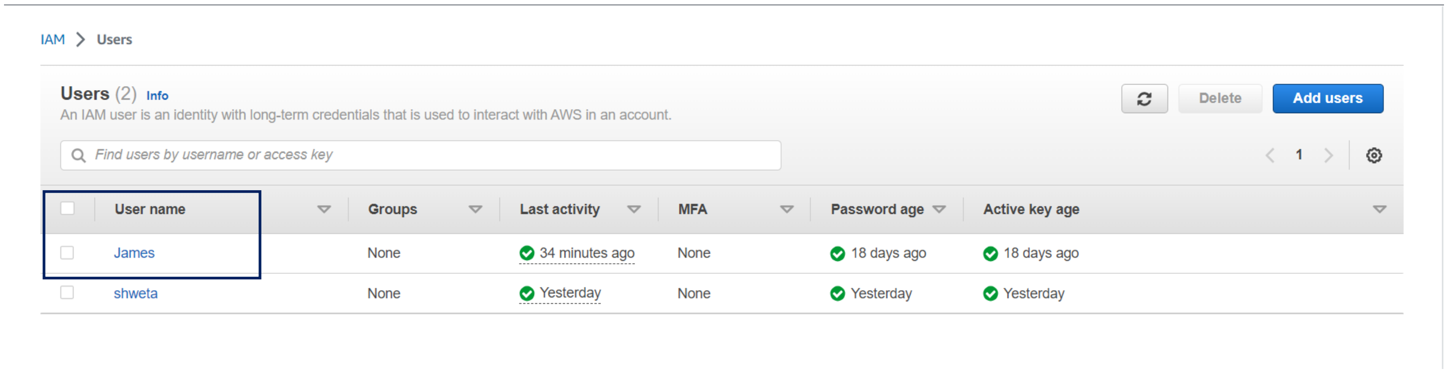
1) First we will create a training dataset in our local system. We will take a min 50-60 images of faces for each person.



This PC > Desktop > img

Name	Date modified	Type	Size
AmitabBachhan	7/6/2023 8:01 AM	File folder	
DonaldTrump	7/6/2023 7:39 AM	File folder	
Krishnakant	7/6/2023 2:42 PM	File folder	
MSDhoni	7/6/2023 7:50 AM	File folder	
NarendraModi	7/6/2023 7:39 AM	File folder	
Shweta	7/6/2023 2:17 PM	File folder	
SRK	7/6/2023 8:10 AM	File folder	

2) In AWS Account we will create one IAM User. And define policy → Do AWS Login using IAM User.

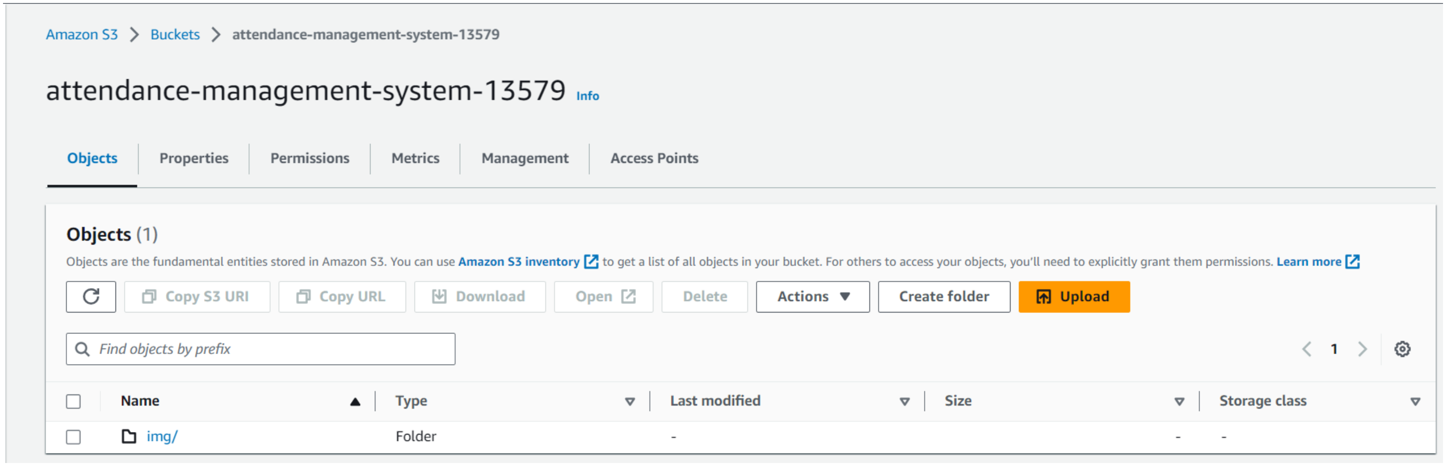


Users (2) Info

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

User name	Groups	Last activity	MFA	Password age	Active key age
James	None	34 minutes ago	None	18 days ago	18 days ago
shweta	None	Yesterday	None	Yesterday	Yesterday

3) Then we will create S3 Bucket in AWS in Region you want → Upload folder of images of our system in S3.

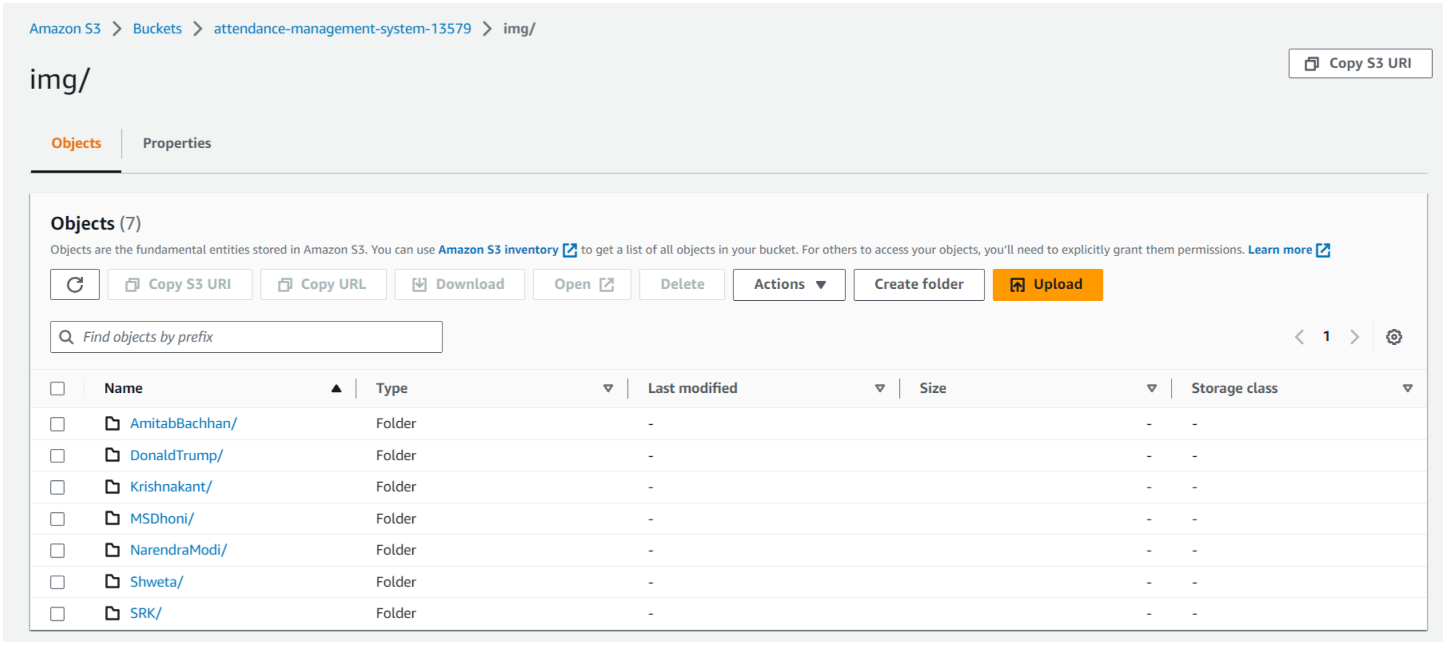

Amazon S3 > Buckets > attendance-management-system-13579

attendance-management-system-13579 Info

Objects (1)

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

Name	Type	Last modified	Size	Storage class
img/	Folder	-	-	-


Amazon S3 > Buckets > attendance-management-system-13579 > img/

img/

Objects (7)

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

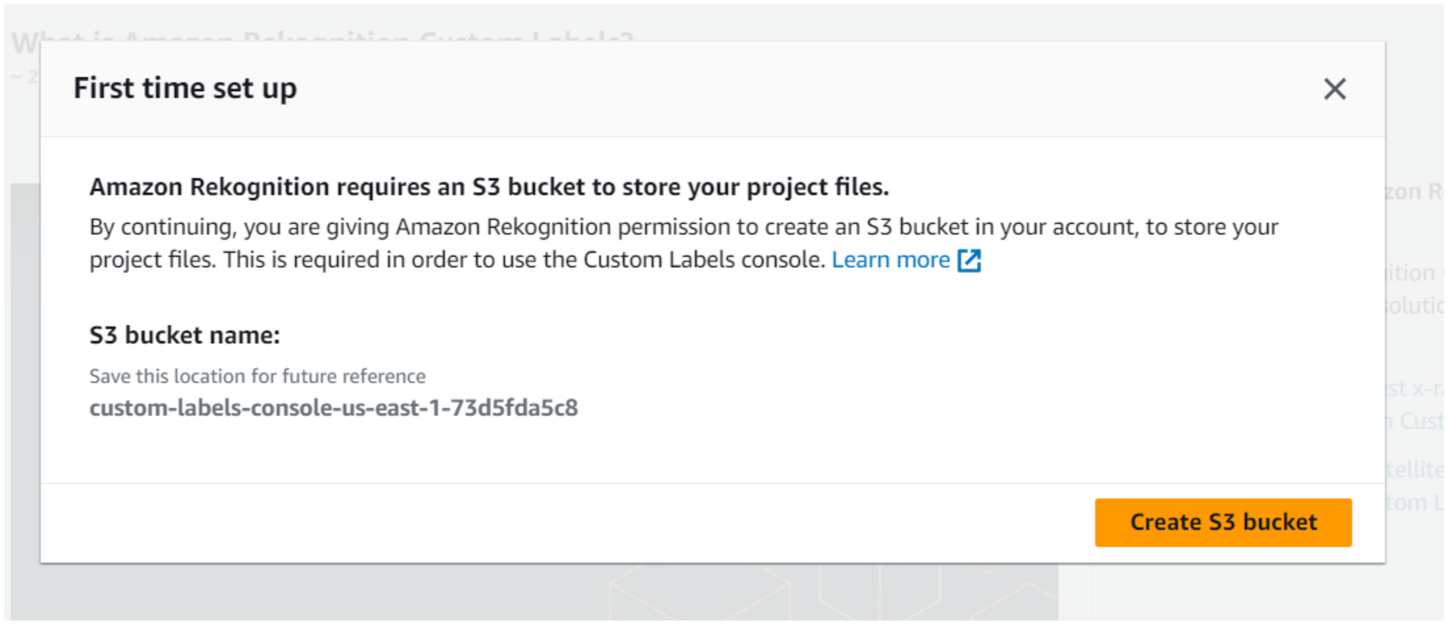
Name	Type	Last modified	Size	Storage class
AmitabBachhan/	Folder	-	-	-
DonaldTrump/	Folder	-	-	-
Krishnakant/	Folder	-	-	-
MSDhoni/	Folder	-	-	-
NarendraModi/	Folder	-	-	-
Shweta/	Folder	-	-	-
SRK/	Folder	-	-	-

4) Go to AWS Rekognition Service in the same region where you have created S3 bucket.

Go to Use Custom Labels → click on Get Started, for first time it will ask you to create S3 bucket with name e.g **custom-labels-console-us-east-1-73d5fda5c8.**

Without this S3 bucket you can't use recognition.

Click on Create S3 bucket.



First time set up

Amazon Rekognition requires an S3 bucket to store your project files.

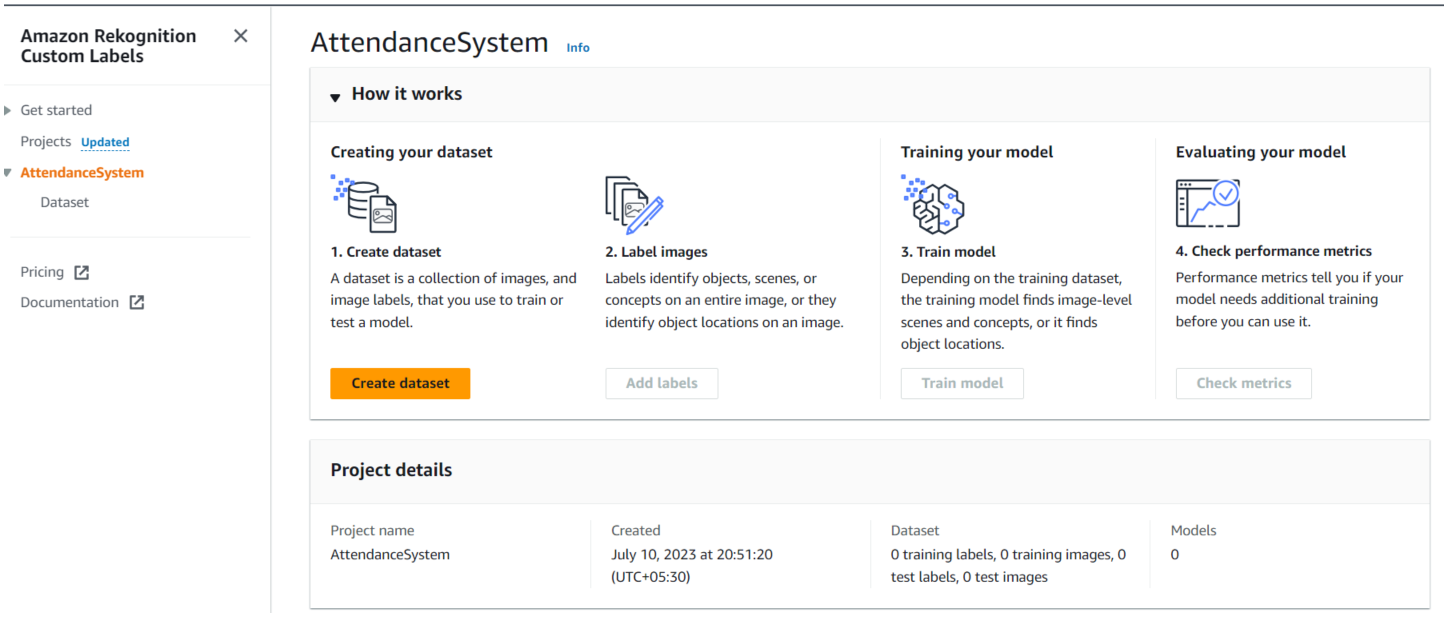
By continuing, you are giving Amazon Rekognition permission to create an S3 bucket in your account, to store your project files. This is required in order to use the Custom Labels console. Learn more

S3 bucket name:

Save this location for future reference

custom-labels-console-us-east-1-73d5fda5c8

5) In Custom Labels → Project → Create Project → give Project Name "AttendanceSystem".



AttendanceSystem Info

How it works

Creating your dataset

1. Create dataset

A dataset is a collection of images, and image labels, that you use to train or test a model.

2. Label images

Labels identify objects, scenes, or concepts on an entire image, or they identify object locations on an image.

Training your model

3. Train model

Depending on the training dataset, the training model finds image-level scenes and concepts, or it finds object locations.

Evaluating your model

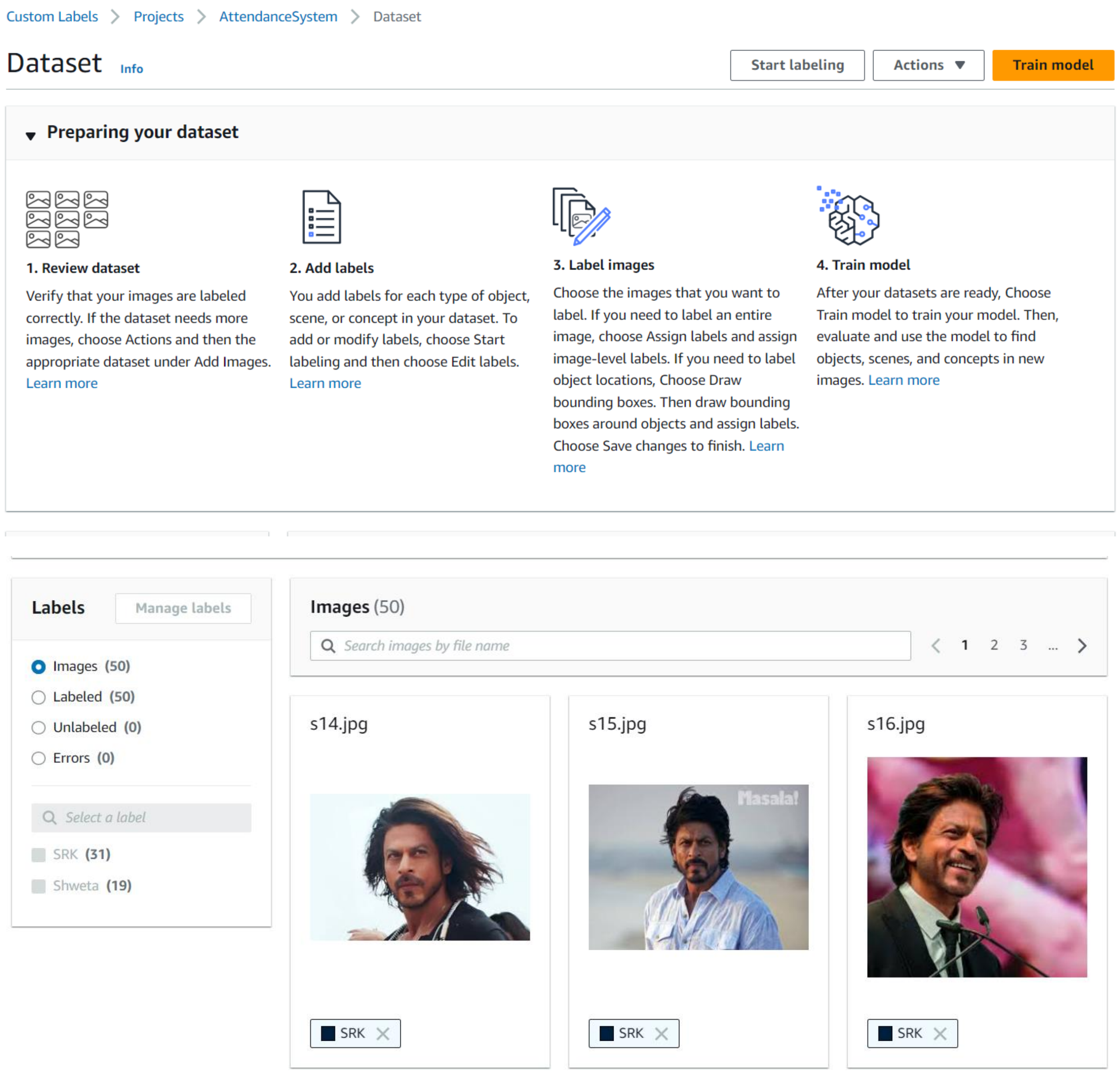
4. Check performance metrics

Performance metrics tell you if your model needs additional training before you can use it.

Project details

Project name	Created	Dataset	Models
AttendanceSystem	July 10, 2023 at 20:51:20 (UTC+05:30)	0 training labels, 0 training images, 0 test labels, 0 test images	0

6) Click on Create Dataset → choose Start with Single Dataset → select Import images from S3 → give URI of S3 bucket (attendance-management-system-13579) → check the box of Automatic labeling → click on Create Dataset.



Custom Labels > Projects > AttendanceSystem > Dataset

Dataset Info

Preparing your dataset

1. Review dataset
Verify that your images are labeled correctly. If the dataset needs more images, choose Actions and then the appropriate dataset under Add Images. Learn more

2. Add labels
You add labels for each type of object, scene, or concept in your dataset. To add or modify labels, choose Start labeling and then choose Edit labels. Learn more

3. Label images
Choose the images that you want to label. If you need to label an entire image, choose Assign labels and assign image-level labels. If you need to label object locations, Choose Draw bounding boxes. Then draw bounding boxes around objects and assign labels. Choose Save changes to finish. Learn more

4. Train model
After your datasets are ready, Choose Train model to train your model. Then, evaluate and use the model to find objects, scenes, and concepts in new images. Learn more

Labels — Manage labels

- [✓ Images (50)]
- Labeled (50)
- Unlabeled (0)
- Errors (0)

Select a label

- SRK (31)
- Shweta (19)

Images (50)

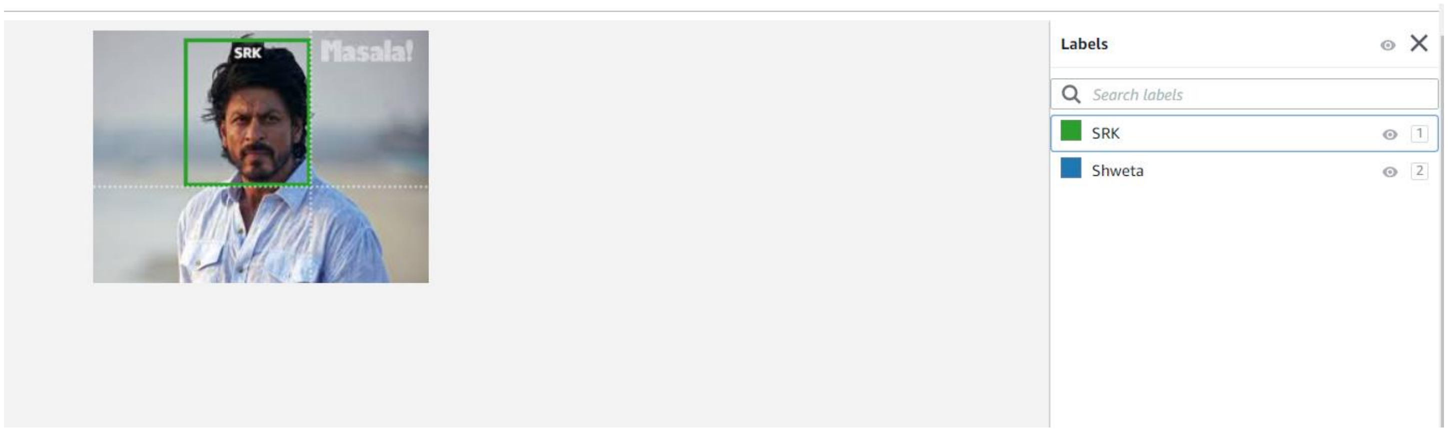
Search images by file name — < 1 2 3 ... >

s14.jpg — SRK

s15.jpg — SRK

s16.jpg — SRK

7) Click on Start Labeling → choose the photos to label → select Draw Bounding Boxes method → click on Done.



Finish labeling.

8) Click on Train Model, it will take 30 mins to 24 hrs to complete the process according to number of images.

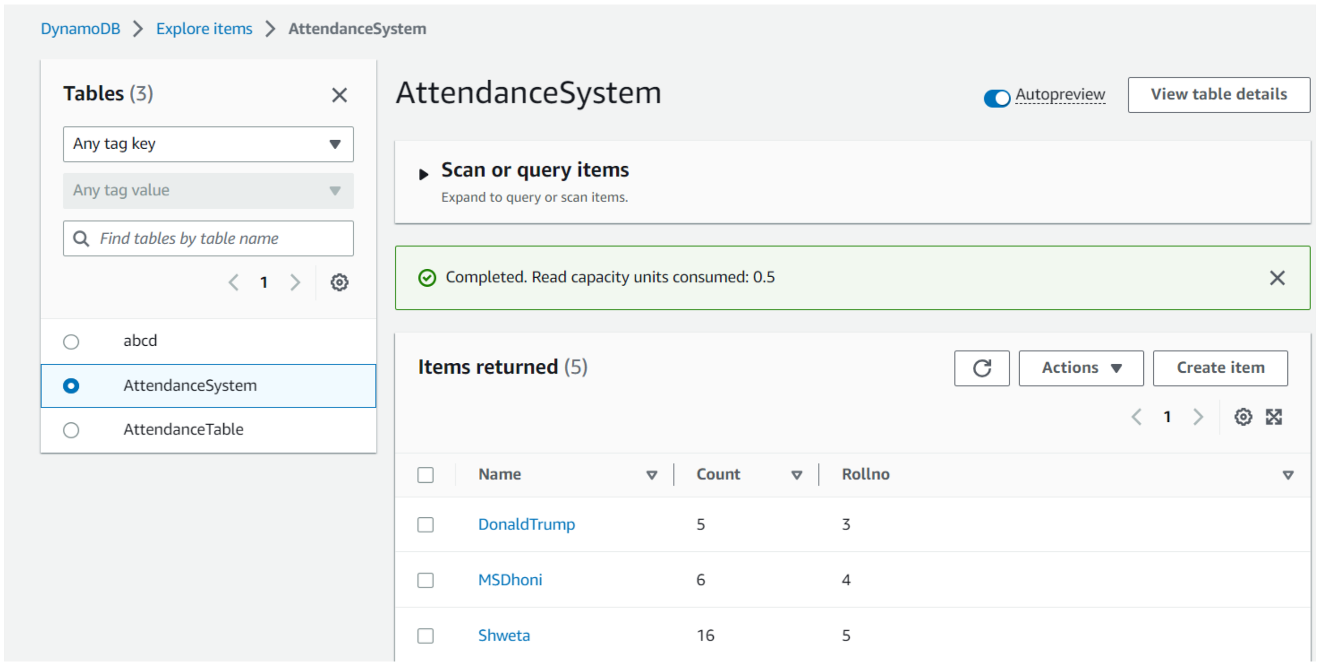
NOTE :

- If you take only one or two persons images then the model will not train. It will show the error "Too few labels in manifest files". So take minimum 5 persons images.
- If you take only 5-15 images for each person then it will not detect you correctly, so you have to take at least 50 images for each person.

Now wait for completing the train model process.

9) Create DynamoDB table.

Give name to table → Give Partition Key as Name → other attributes ‘Rollno’ and ‘Count’. And add the name and rollno in table. Keep Count as 0.



Completed. Read capacity units consumed: 0.5

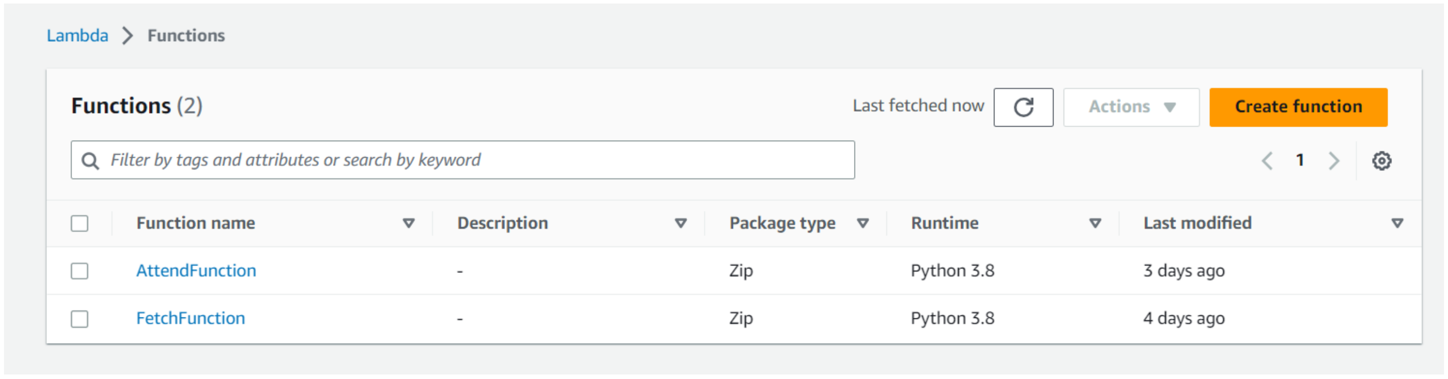
Items returned (5)

Name	Count	Rollno
DonaldTrump	5	3
MSDhoni	6	4
Shweta	16	5

10) Now we will create Lambda function and API Gateway in same region.

- <u>For getting attendance from captured img and adding into dynamo db.</u>

Go to Lambda → create Function → Author from Scratch → give Function Name → select Runtime Python 3.10 and all other settings will be default → click on create function.



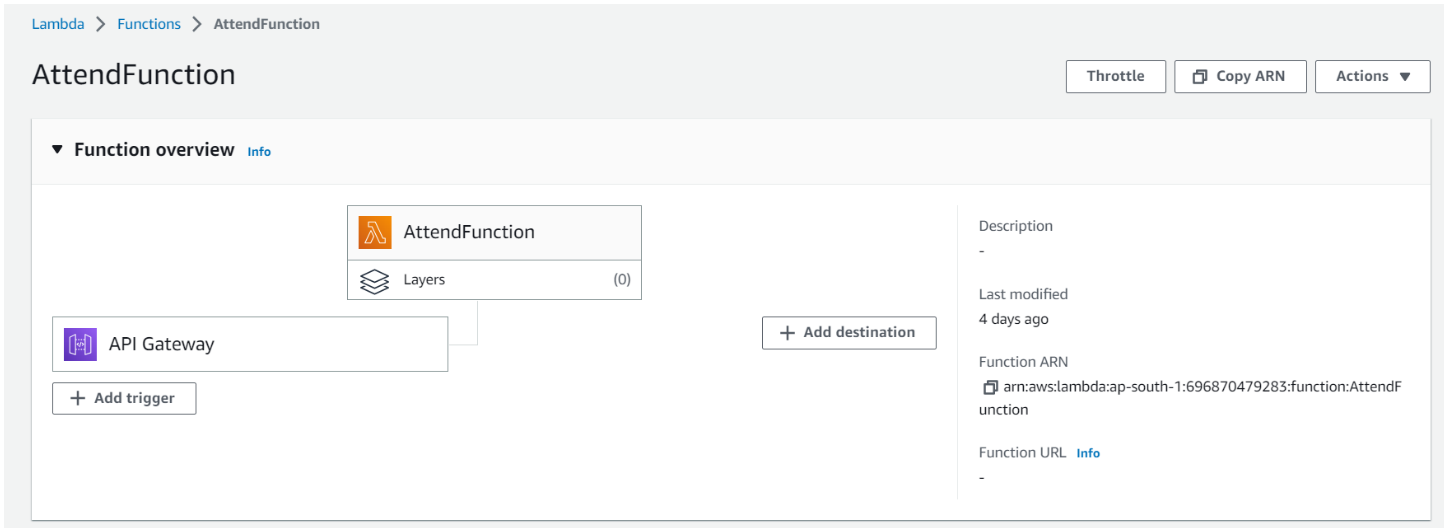
Functions (2)

Function name	Description	Package type	Runtime	Last modified
AttendFunction	-	Zip	Python 3.8	3 days ago
FetchFunction	-	Zip	Python 3.8	4 days ago

To integrate this Lambda Function with API Gateway :

In Lambda Function Overview → Add Trigger → select a source – API Gateway → choose new API → choose a Rest API → security type Open → click on Add.

Your API will be created and automatically attached with lambda.



Description
-

Last modified
4 days ago

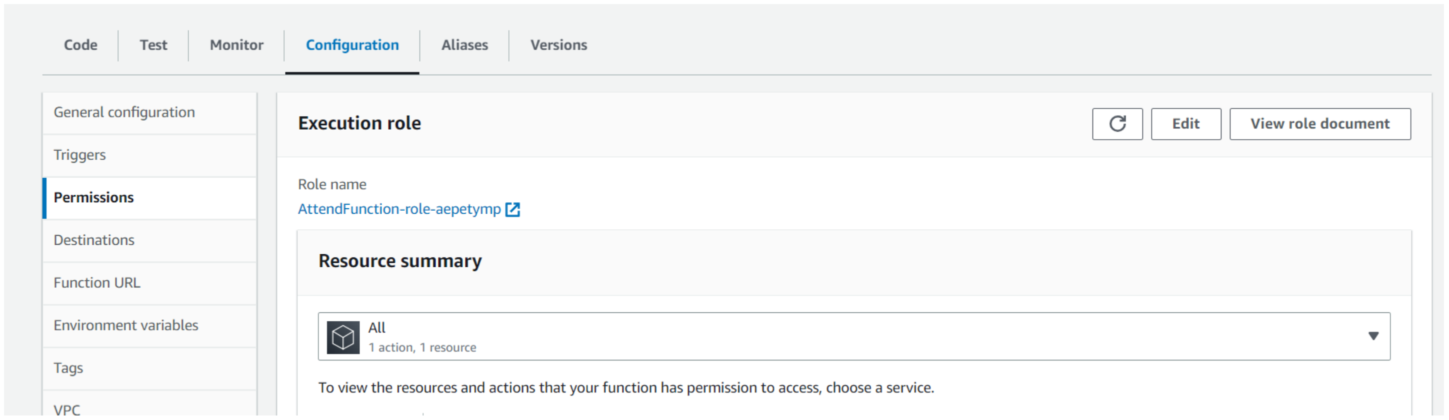
Function ARN
arn:aws:lambda:ap-south-1:696870479283:function:AttendFunction

Function URL Info
-

Now in Lambda go to Configuration tab in that → Permissions

There role name is given.

Click on that role name → it will open in IAM Services → Add permissions → Give permission for full access of Dynamodb or All administrator access.

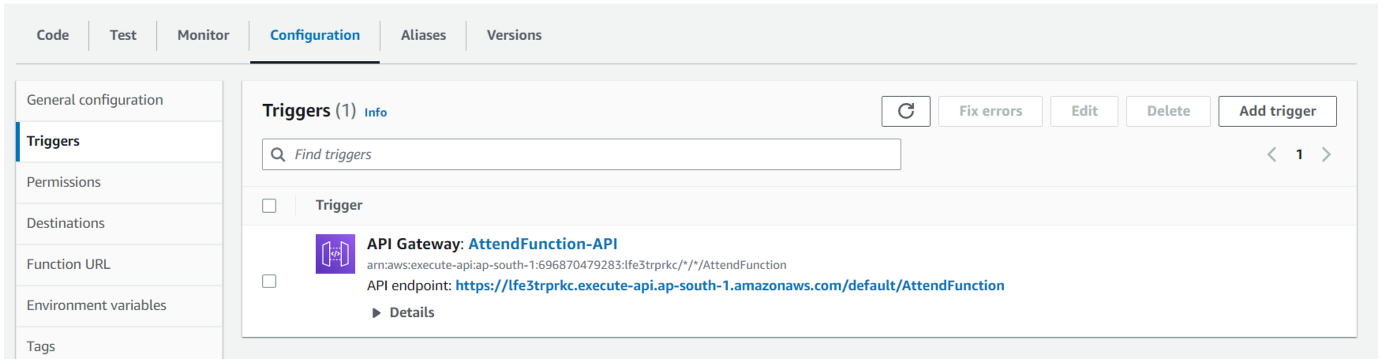


Role name
AttendFunction-role-aepetymp

All
1 action, 1 resource

To view the resources and actions that your function has permission to access, choose a service.

In the configuration tab, the endpoint of your API gateway is given.

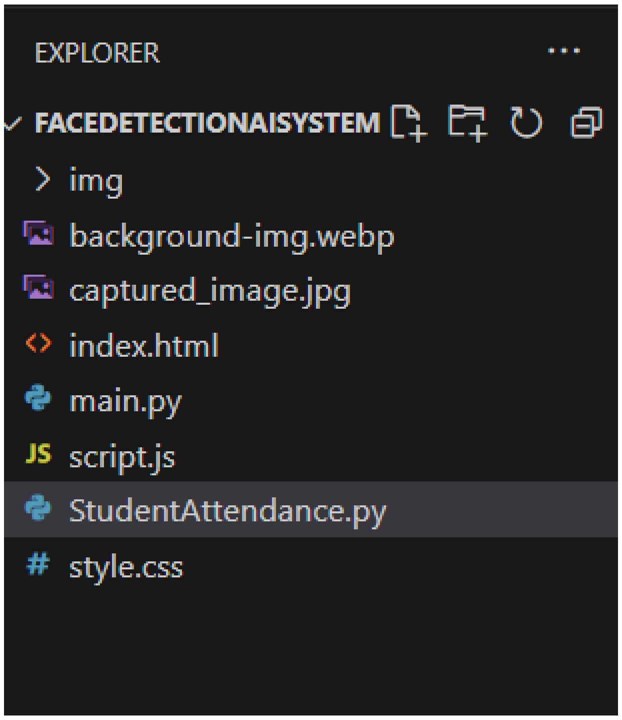


API Gateway: AttendFunction-API
arn:aws:execute-api:ap-south-1:696870479283:lfe3trprkc/*/*/AttendFunction
API endpoint: https://lfe3trprkc.execute-api.ap-south-1.amazonaws.com/default/AttendFunction

Code :

Now to capture a photo of person and add there's attendance in in Dynamodb via API Gateway using Lambda function, we have written one Python Code StudentAttendance.py, main.py and one lambda function (AttendFunction) code.

Project Folder Directory, you also have to create one folder for image i.e. img.



```
EXPLORER
FACEDETECTIONAISYSTEM
  > img
  background-img.webp
  captured_image.jpg
  index.html
  main.py
  script.js
  StudentAttendance.py
  style.css
```

- *StudentAttendance.py :*

```
import boto3
import requests
import datetime
import time
import cv2

#Credentials--------------------------------------------------------------------------------
-
client = boto3.client('rekognition',
                    aws_access_key_id="AKIA2EQFXUWZWLJG6WTR",
                    aws_secret_access_key="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",
                    region_name='ap-south-1')



#Capture images for every 1 hour and store the image with current date and time -------------
----------------------------------------------------------------------
```

```python
for j in range(0, 6):
    current_time = datetime.datetime.now().strftime("%d-%m-%y  %H-%M-%S ")
    print(current_time)
    camera = cv2.VideoCapture(0)



    while True:
        # Capture the video frame by frame
        ret, frame = camera.read()

        # Display the resulting frame
        cv2.imshow('frame', frame)
        # Check if the image needs to be captured
        if cv2.waitKey(1) & 0xFF == ord(' '):
            # Save the captured frame as an image
            cv2.imwrite('img/' + current_time + '.jpg', frame)
            print("Image captured!")
            # Reset the flag
            break

        # Check if the 'q' button is pressed to quit
        if cv2.waitKey(1) & 0xFF == ord('q'):
            exit()


    del (camera)

#Send the captured image to AWS S3 Bucket----------------------------------------------------
----------------------------------
    clients3 = boto3.client('s3', aws_access_key_id="AKIA2EQFXUWZWLJG6WTR",
                  aws_secret_access_key="05K61SpmHdfXk1Jt0BSxz8jZihF2MezaGBQWN+s1",
region_name='ap-south-1')
    # clients3.upload_file("Hourly Class Images/"+current_time+'.jpg', 'add your S3 bucket
name', current_time+'.jpg')

    clients3.upload_file("img/" + current_time + '.jpg', 'attendance-management-system-
13579', current_time + '.jpg')


    #Recoginze students in captured image ---------------------------------------------------
------------------------------------
    image_path = 'img/' + current_time + '.jpg'
    with open(image_path,'rb') as source_image:
        source_bytes = source_image.read()
    print(type(source_bytes))

    print("Recognition Service")
    response = client.detect_custom_labels(
 #Add your recognition project arn----------------------------------------------------------
--------------------------------------
        ProjectVersionArn='arn:aws:rekognition:ap-south-
1:696870479283:project/AttendanceSystem/version/AttendanceSystem.2023-07-
06T15.49.54/1688638795363',


        Image={
            'Bytes': source_bytes
        },

    )

    print(response)
    if not len(response['CustomLabels']):
         print('Not identified')

    else:
        str = response['CustomLabels'][0]['Name']
        print(str)

      # Update the attendance of recognized student in DynamoDB by calling the API
      Add API endpoint--------------------------------------------------------------

        url = "https://1fe3trprkc.execute-api.ap-south-
1.amazonaws.com/default/AttendFunction?Name=" + str

        resp = requests.get(url)
        print("Success ")
        if resp.status_code==200:
            print("Success")

    time.sleep(3600)
```

NOTE :

- In the above code you have to edit your Rekognition Project ARN and API gateway URL.
- Change your IAM user credentials and region name.
- Change S3 bucket name and path of images folder.

- ***main.py :***

```python
# import the opencv library
import cv2

# define a video capture object
vid = cv2.VideoCapture(0)

# flag to indicate whether to capture an image
capture_image = False

while True:
    # Capture the video frame by frame
    ret, frame = vid.read()

    # Display the resulting frame
    cv2.imshow('frame', frame)
    # Check if the image needs to be captured
    if cv2.waitKey(1) & 0xFF == ord(' '):
        # Save the captured frame as an image
        cv2.imwrite('captured_image.jpg', frame)
        print("Image captured!")
        # Reset the flag
        capture_image = False

    # Check if the 'q' button is pressed to quit
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release the video capture object
vid.release()
# Destroy all the windows
cv2.destroyAllWindows()
```

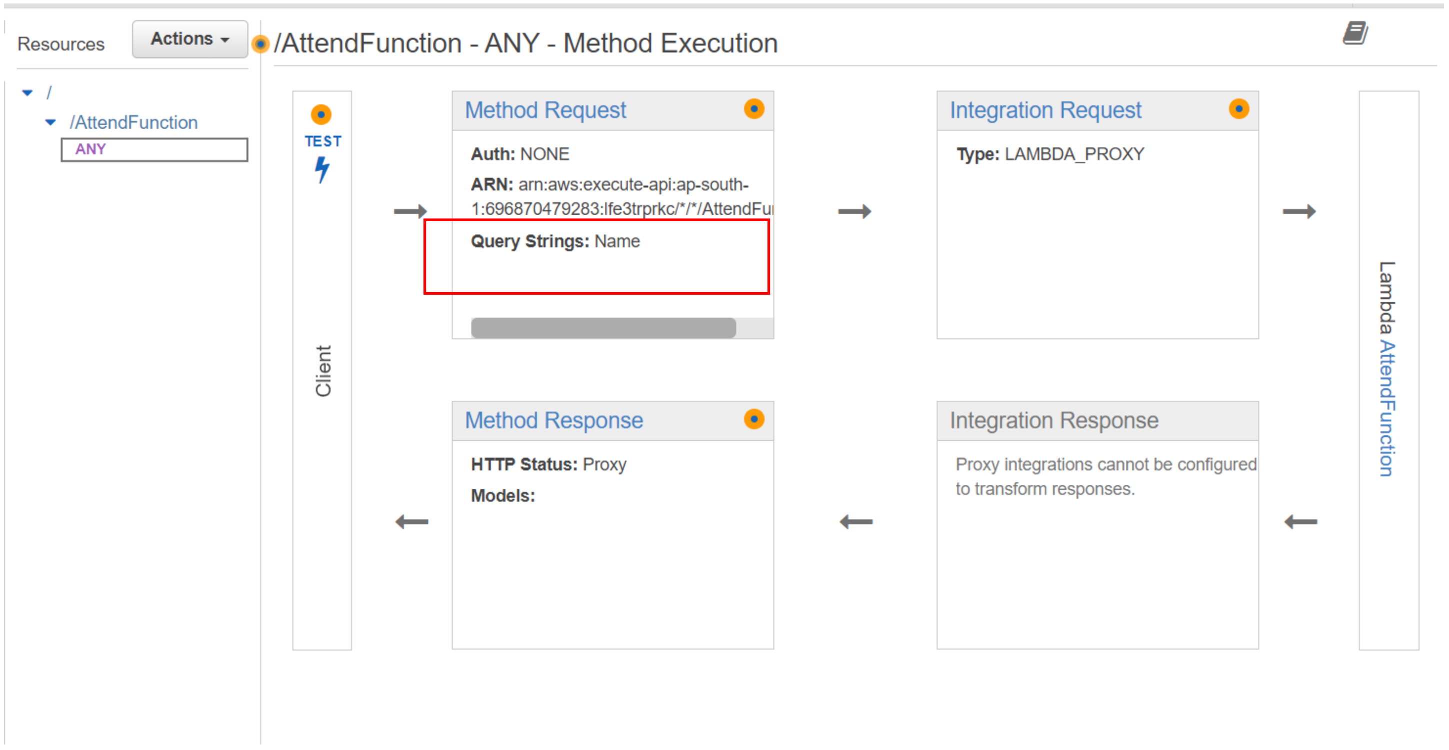
LambdaFunction :

```
import json
import boto3

dynamo=boto3.resource("dynamodb")

table=dynamo.Table("AttendanceSystem")
def lambda_handler(event, context):
    # TODO implement

    # Data was not updated in dynamodb bccoz (queryStringParameter) was not not there..
    res = table.get_item(Key={"Name": event['queryStringParameters']['Name']})
    print(res['Item']['Name'])
    Count = res['Item']['Count']
    Count= Count+1
    inp = {"Rollno":res["Item"]['Rollno'],"Count":Count, "Name":res['Item']['Name']}
    table.put_item(Item=inp)
    return "Successful"
```

NOTE :

- Change dynamodb table name.
- Change your dynamo db attributes name.
- And add that queryStringParameter as name in REST API also.

(If we remove this from REST API and Lambda function then the data will not added in dynamodb table.)



Auth: NONE

ARN: arn:aws:execute-api:ap-south-1:696870479283:lfe3trprkc/*/*/AttendFu

Query Strings: Name

Type: LAMBDA_PROXY

HTTP Status: Proxy

Models:

Proxy integrations cannot be configured to transform responses.

<u>B] For Fetching Data from Dynamodb on Web Application :</u>

1) Create on Web Application using html and js.

- *<u>index.html :</u>*

```html
<!DOCTYPE html>
<html>
<head>
  <title>Attendance Table</title>
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div class="container mt-5">
    <table id="attendanceTable" class="table styled-table ">
      <thead>
        <tr>
          <th>Name</th>
          <th>Rollno</th>
          <th>Attendance Count</th>
        </tr>
      </thead>
      <tbody></tbody>
    </table>
  </div>

  <script src="script.js"></script>
</body>
</html>
```

- *<u>style.css :</u>*

```css
*{
    padding: 0;
    margin: 0;
}
body {
    background: url("background-img.webp") no-repeat center center fixed;
    background-size: cover;

  }
.styled-table {
    border-collapse: collapse;
    margin: 25px 0;
    font-size: 0.9em;
    font-family: sans-serif;
    min-width: 400px;
    box-shadow: 0 0 20px rgba(0, 0, 0, 0.15);
}

.styled-table thead tr {
    background-color: #93A0FE;
    color: #ffffff;
    text-align: left;
}

.styled-table th,
.styled-table td {
    padding: 12px 15px;
}

/* .styled-table tbody tr {
    border-bottom: 1px solid #E5606A;
} */

.styled-table tbody tr:nth-of-type(even) {
    background-color: #FFFFFF;
}

.styled-table tbody tr:nth-of-type(odd) {
    background-color: #E4E4E4;
}


/* .styled-table tbody tr:last-of-type {
    border-bottom: 1px solid #009879;
} */

.styled-table tbody tr.active-row {
    font-weight: bold;
    background-color: #009879;
}
```

- *script.js :*

```
document.addEventListener('DOMContentLoaded', () => {
    fetch('https://x26zhzj86f.execute-api.ap-south-1.amazonaws.com/default/FetchFunction')
      .then(response => response.json())
      .then(data => {
        const tableBody = document.querySelector('#attendanceTable tbody');
        data.forEach(student => {
          const row = document.createElement('tr');
          const rollNoCell = document.createElement('td');
          const nameCell = document.createElement('td');
          const attendanceCell = document.createElement('td');

          rollNoCell.textContent = student.Name;
          nameCell.textContent = student.Rollno;
          attendanceCell.textContent = student.Count;

          row.appendChild(rollNoCell);
          row.appendChild(nameCell);
          row.appendChild(attendanceCell);
          tableBody.appendChild(row);
        });
      })
      .catch(error => {
        console.error(error);
      });
  });
```

NOTE :

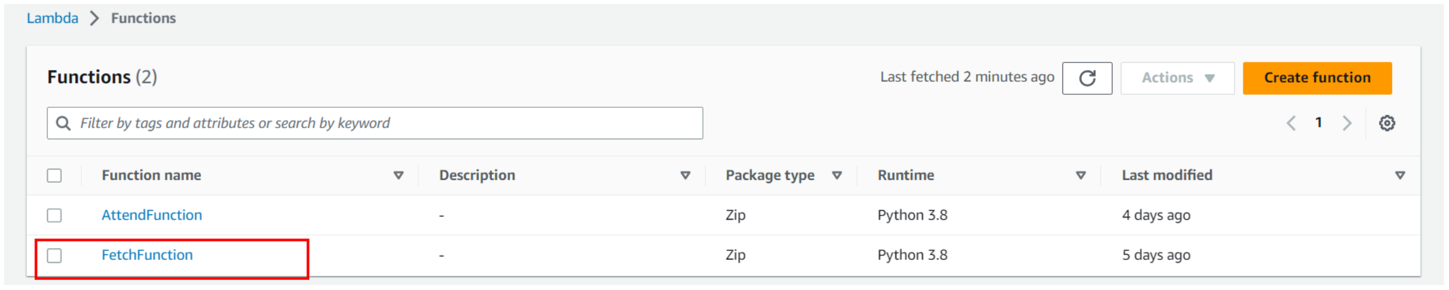
- In script.js code you have to add second API URL. i.e. FetchFunction
- This API we will create in next further steps.

2) For fetching data on created web application we will use API gateway.

This API gateway will take the data from dynamodb using Lambda Function.

<u>So first we will create Lambda Function to get data from dynamodb.</u>

Go to Lambda → create Function → Author from Scratch → give Function Name → select Runtime Python 3.10 and all other settings will be default → click on create function.

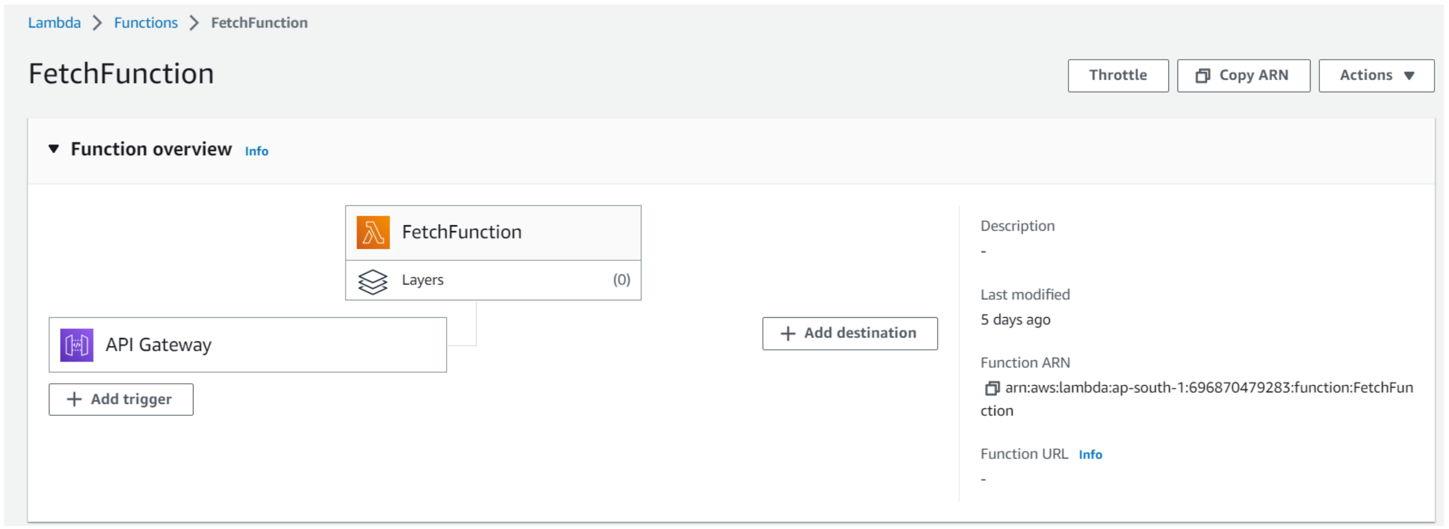

Lambda > Functions
Functions (2)
Last fetched 2 minutes ago
Actions
Create function
Filter by tags and attributes or search by keyword

Function name	Description	Package type	Runtime	Last modified
AttendFunction	-	Zip	Python 3.8	4 days ago
FetchFunction	-	Zip	Python 3.8	5 days ago

3) To integrate this Lambda Function with API Gateway :

In Lambda Function Overview → Add Trigger → select a source – API Gateway → choose new API → choose a HTTP API → security type Open → click on Add.

Your API will be created and automatically attached with lambda.



Description
-

Last modified
5 days ago

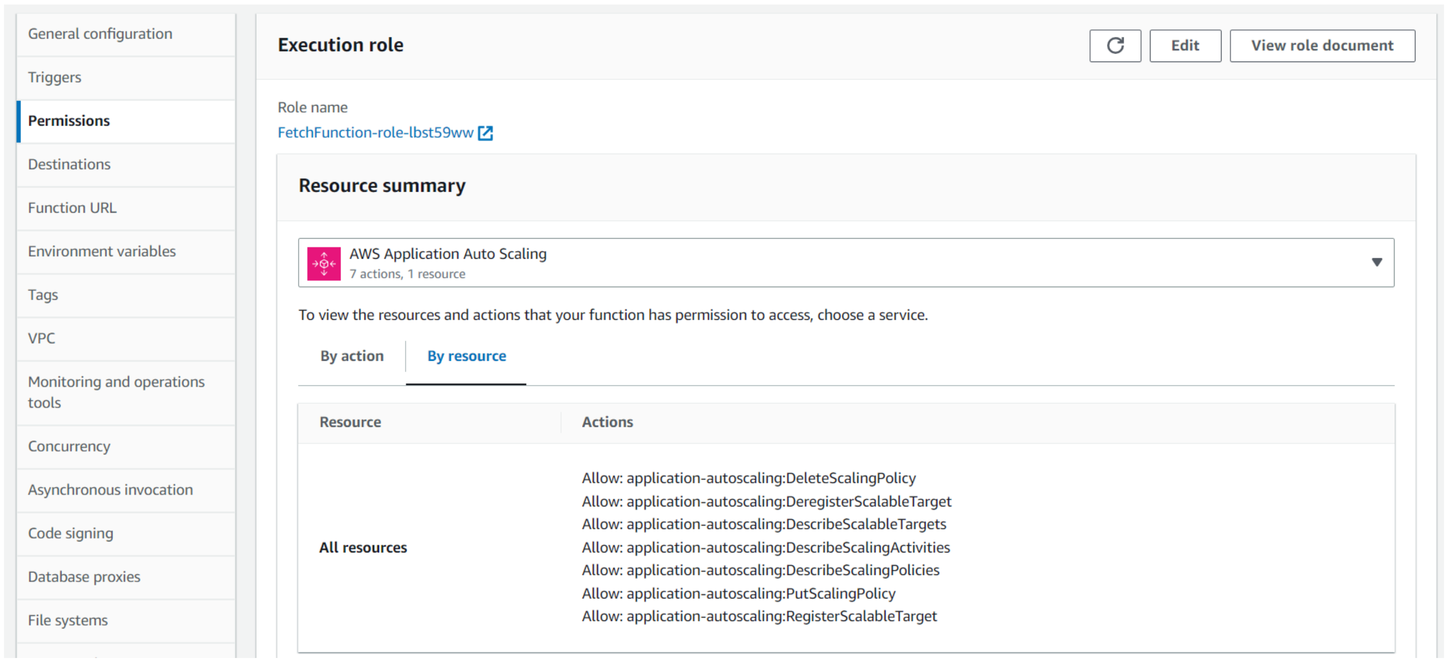
Function ARN
arn:aws:lambda:ap-south-1:696870479283:function:FetchFunction

Function URL Info
-

Now in Lambda go to Configuration tab in that → Permissions

There role name is given.

Click on that role name → it will open in IAM Services → Add permissions → Give permission for full access of Dynamodb or All administrator access.



Execution role

Role name
FetchFunction-role-lbst59ww

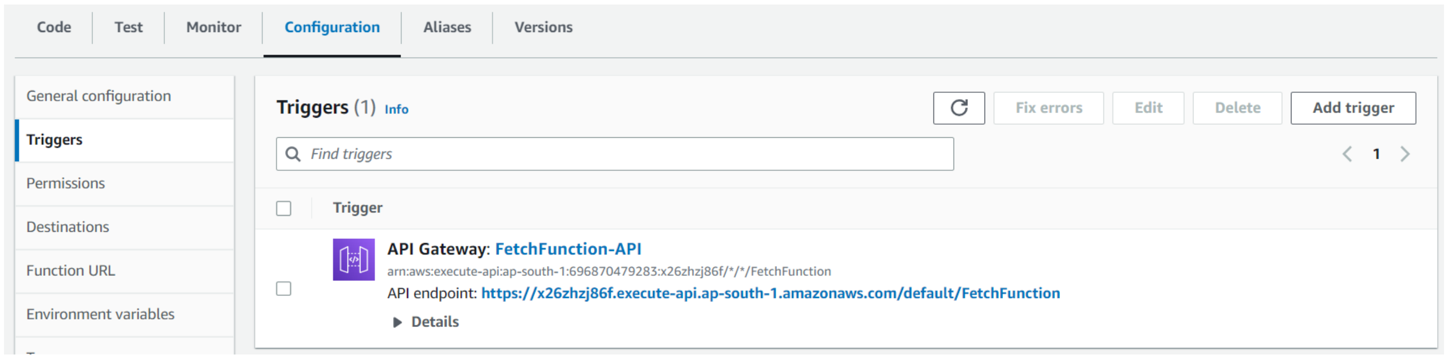
Resource summary

AWS Application Auto Scaling
7 actions, 1 resource

To view the resources and actions that your function has permission to access, choose a service.

Resource	Actions
All resources	Allow: application-autoscaling:DeleteScalingPolicy Allow: application-autoscaling:DeregisterScalableTarget Allow: application-autoscaling:DescribeScalableTargets Allow: application-autoscaling:DescribeScalingActivities Allow: application-autoscaling:DescribeScalingPolicies Allow: application-autoscaling:PutScalingPolicy Allow: application-autoscaling:RegisterScalableTarget

In the configuration tab, the endpoint of your API gateway is given.

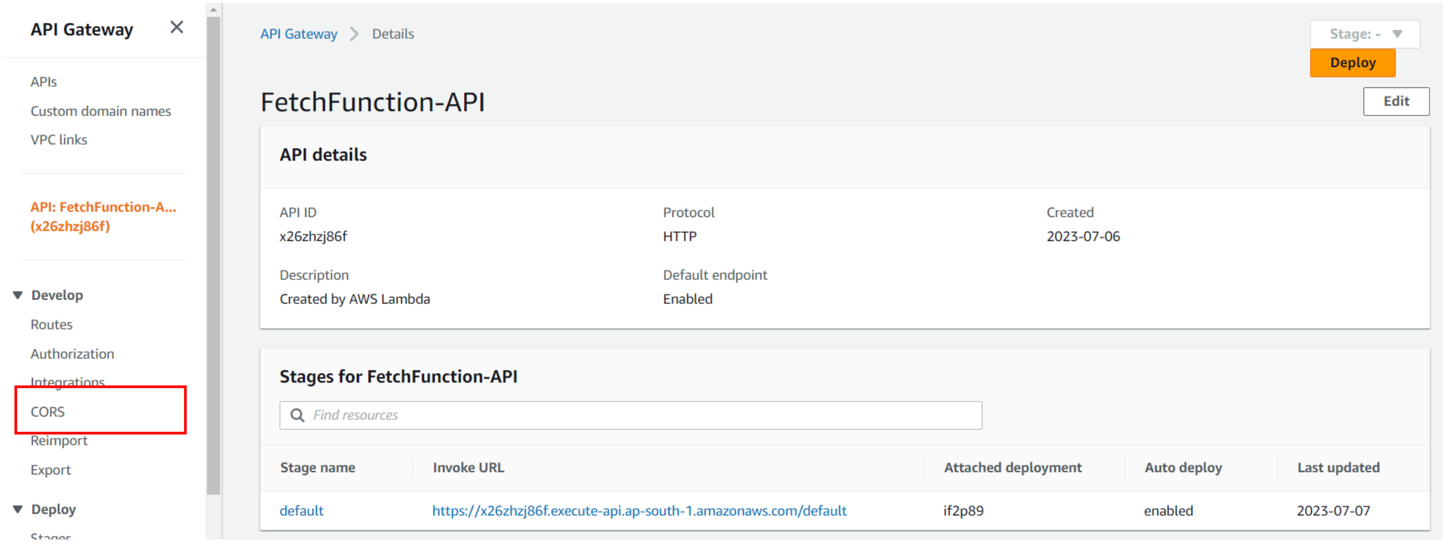


Triggers (1)

Trigger

API Gateway: FetchFunction-API
arn:aws:execute-api:ap-south-1:696870479283:x26zhzj86f/*/*/FetchFunction
API endpoint: https://x26zhzj86f.execute-api.ap-south-1.amazonaws.com/default/FetchFunction

Now go to the API Services → click on your API → here go to the CORS option.



FetchFunction-API

API details

API ID
x26zhzj86f

Protocol
HTTP

Created
2023-07-06

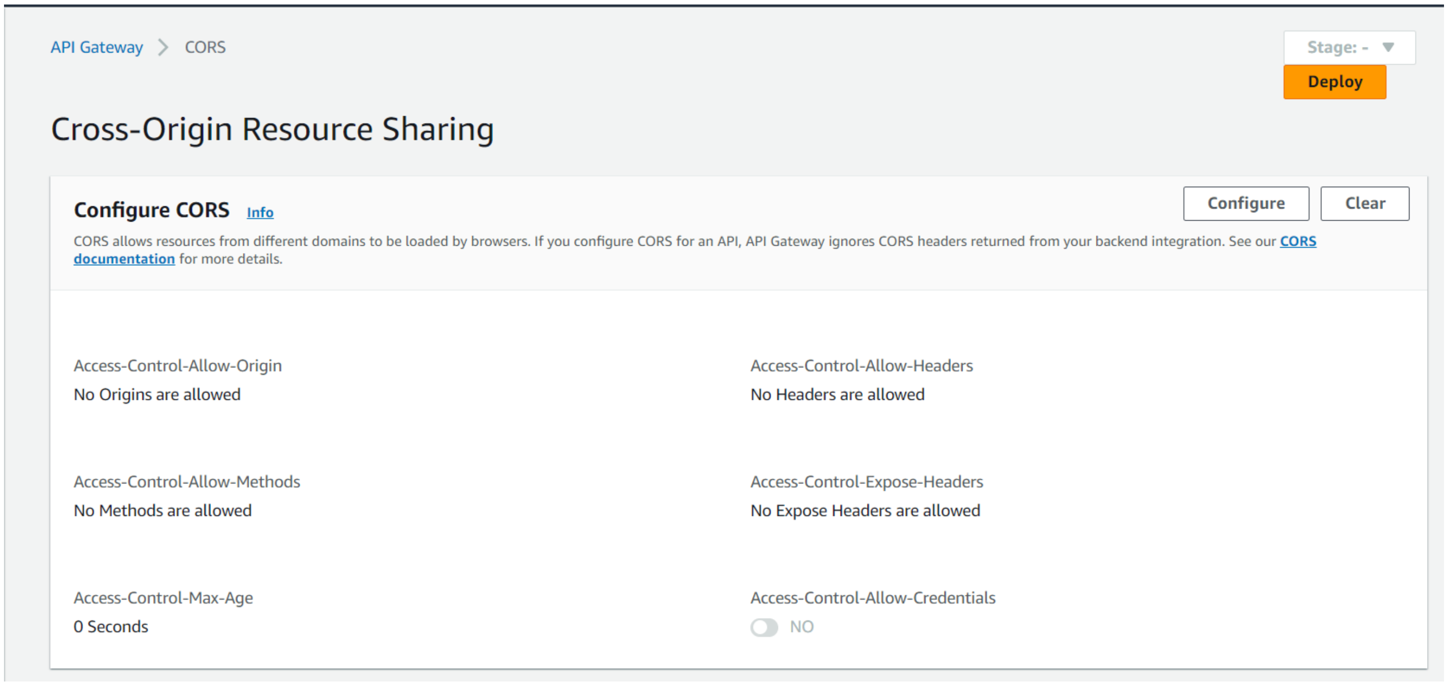
Description
Created by AWS Lambda

Default endpoint
Enabled

Stages for FetchFunction-API

Stage name	Invoke URL	Attached deployment	Auto deploy	Last updated
default	https://x26zhzj86f.execute-api.ap-south-1.amazonaws.com/default	if2p89	enabled	2023-07-07

It will be look like



Cross-Origin Resource Sharing

Configure CORS

CORS allows resources from different domains to be loaded by browsers. If you configure CORS for an API, API Gateway ignores CORS headers returned from your backend integration. See our CORS documentation for more details.

Access-Control-Allow-Origin
No Origins are allowed

Access-Control-Allow-Headers
No Headers are allowed

Access-Control-Allow-Methods
No Methods are allowed

Access-Control-Expose-Headers
No Expose Headers are allowed

Access-Control-Max-Age
0 Seconds

Access-Control-Allow-Credentials
NO

NOTE :

- If while fetching data on web application if data is not shown on web page, then in the above CORS Configuration you can configure * everywhere then save.
- Sometime it need to clear then also data will be fetch.

4) Add code of the Lambda Function.

Code :

```
import json
import boto3

dynamo=boto3.resource("dynamodb")
table=dynamo.Table("AttendanceSystem")
def lambda_handler(event, context):
    # TODO implement
    response=table.scan()
    print(response)
    # items = response['Items']
    # print(items)
    # print(items)
    data= []
    for item in response['Items']:
      item['Rollno'] = str(item['Rollno'])
      item['Count'] = str(item['Count'])
      data.append(item)
    print(data)
    return{  "statusCode": 200,
    "headers": {
      'Access-Control-Allow-Origin': '*',
      'Access-Control-Allow-Credentials':True,
      'Access-Control-Allow-Methods': 'GET, POST, PUT, DELETE',
      'Access-Control-Allow-Headers': 'Content-Type, Authorization',
    },
    "body": json.dumps(response["Items"])}
```

NOTE :

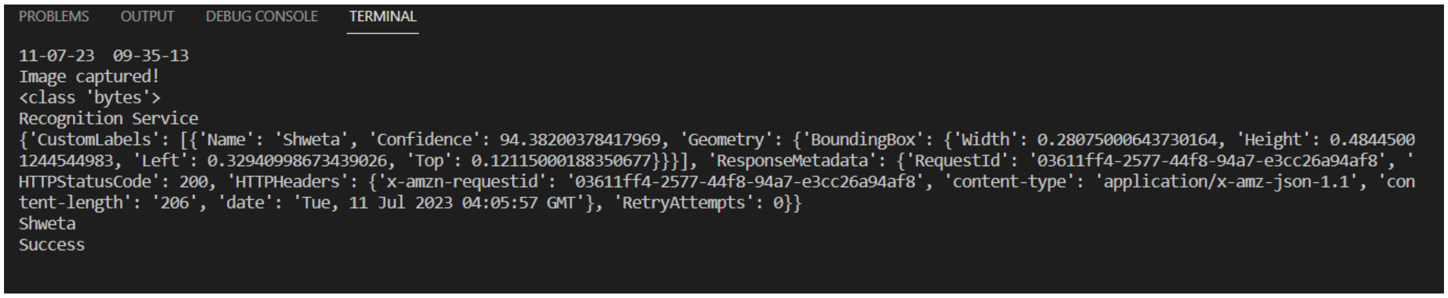
- Change your dynamodb table name and attributes.

Here all steps are completed for Proejct.

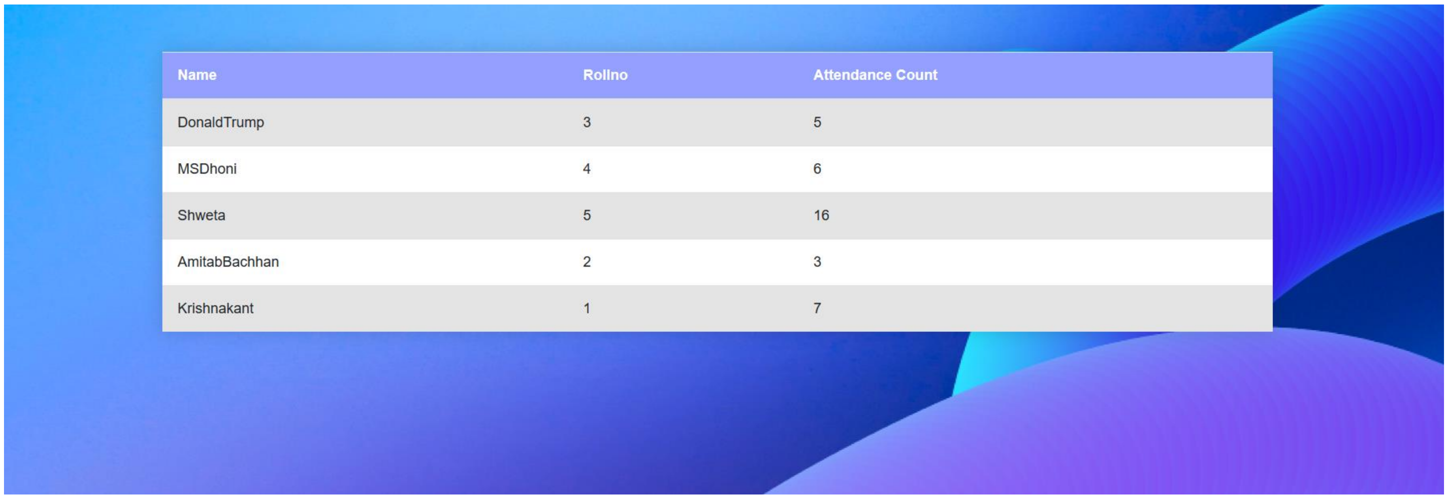
Output :

I will run Python code, then camera will on → capture image by clicking space button.

Person’s name will be shown in terminal.



```
11-07-23  09-35-13
Image captured!
<class 'bytes'>
Recognition Service
{'CustomLabels': [{'Name': 'Shweta', 'Confidence': 94.38200378417969, 'Geometry': {'BoundingBox': {'Width': 0.28075000643730164, 'Height': 0.4844500
1244544983, 'Left': 0.32940998673439026, 'Top': 0.12115000188350677}}}], 'ResponseMetadata': {'RequestId': '03611ff4-2577-44f8-94a7-e3cc26a94af8', '
HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid': '03611ff4-2577-44f8-94a7-e3cc26a94af8', 'content-type': 'application/x-amz-json-1.1', 'con
tent-length': '206', 'date': 'Tue, 11 Jul 2023 04:05:57 GMT'}, 'RetryAttempts': 0}}
Shweta
Success
```



Name	Rollno	Attendance Count
DonaldTrump	3	5
MSDhoni	4	6
Shweta	5	16
AmitabBachhan	2	3
Krishnakant	1	7

Future Scope

The world is using facial recognition technology and enjoying its benefits. The technology and its applications can be applied across different segments in the country.

● Preventing the frauds at ATMs in India. A database of all customers with ATM cards in India can be created and facial recognition systems can be installed. So, whenever a user enters an ATM his photograph will be taken to permit access after it is matched with a stored photo from the database.

● Passport and visa verification can also be done using this technology.

● Also, driving license verification can be done using the same approach.

● It can also be used during examinations such as Civil Services Exam, SSC, IIT, MBBS, and others to identify the candidates.

● This system can be deployed for verification and attendance tracking at various government offices and corporates.