### LAMBDA FUNCTION -> **imagesS3Bucket\_R**

API URL ->

https://j0mmgihtaj.execute-api.us-east-1.amazonaws.com/v1/imagesS3Bucket\_R

Resource -> **imagesS3Bucket\_R**

Stage -> v1

### 1. Role of the API

This API uploads images to an AWS S3 bucket based on the provided imageType and image data.

### 2. Functioning

* **Token Verification**: Validates the JWT token to authenticate the request.
* **CORS Handling**: Sets appropriate headers to handle Cross-Origin Resource Sharing (CORS).
* **Request Validation**: Ensures required fields like imageType and image are present and valid.
* **Image Upload**: Uploads the image to the specified folder in the S3 bucket.
* **Response Handling**: Returns success or error messages based on the operation's outcome.

### 3. Request Body

**Headers:**

Authorization

Value: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiI2NjcxYWU1NTZhNWY0YTRjNWNhMzMzYjUiLCJlbWFpbCI6InNhaW5pcHM5NDE0NjZAZ21haWwuY29tIiwiaWF0IjoxNzIyMDg1MzgwLCJleHAiOjE3MjIxMDMzODB9.7E2krmYl5RT81t4zLaPpuuBY5M81zvUiQeOwO2kTF8w

**Body:**

{

"imageType": "option",

"image": "Base\_64\_image"

}

Please Note -> imageType is Flag,

· -> We are using 4 flag, -> profile', 'question', 'option', 'answer

· IF any is not used any then it will count in default.

· ImageType & image are required payload

· Please make sure image will passed in payload as base\_64\_encoded

### 4. Response

**Success Response:**

* **Status Code**: 200
* **Body**: {"message": "Image uploaded successfully", "imageUrl": "URL\_of\_uploaded\_image"}

**Error Responses:**

* **Status Code**: 401
  + **Body**: {"message": "Unauthorized: No token provided"}
  + **Body**: {"message": "Unauthorized: Invalid token"}
* **Status Code**: 400
  + **Body**: {"message": "Bad Request: Invalid JSON"}
  + **Body**: {"message": "Bad Request: imageType and image are required"}
* **Status Code**: 500
  + **Body**: {"message": "Internal Server Error: Failed to upload image"}

### 5. Logic

1. **Token Verification**: Validates the JWT token to ensure the user is authorized to perform the operation.
2. **CORS Headers**: Sets headers to allow CORS requests from any origin.
3. **Request Validation**: Ensures required fields like imageType and image are present and valid.
4. **Image Upload**: Uploads the image to the specified folder in the S3 bucket.
5. **Response Creation**: Generates and returns appropriate responses based on the success or failure of the operation.

### 6. Dependencies

* **aws-sdk**: AWS SDK for JavaScript.
* **jsonwebtoken**: Library for generating and verifying JSON Web Tokens (JWT).

CODE:

const AWS = require('aws-sdk');

const jwt = require('jsonwebtoken');

const s3 = new AWS.S3();

const jwtSecret = process.env.JWT\_SECRET\_KEY; // Replace with your JWT secret key

const bucketName = process.env.S3\_BUCKET\_NAME; // Replace with your S3 bucket name

exports.handler = async (event, context) => {

const preflightResponseHeaders = {

"Access-Control-Allow-Origin": "https://student.exambuilder.online", // CORS header

"Access-Control-Allow-Methods": "OPTIONS,POST",

"Access-Control-Allow-Headers": "Content-Type, Authorization"

};

const responseHeaders = {

"Access-Control-Allow-Origin": "https://student.exambuilder.online", // CORS header

};

if (event.httpMethod === 'OPTIONS') {

return {

statusCode: 200,

headers: preflightResponseHeaders,

body: JSON.stringify({ message: 'Preflight check successful' })

};

}

const token = event.headers.Authorization;

if (!token) {

return {

statusCode: 401,

headers: responseHeaders,

body: JSON.stringify({ message: 'Unauthorized: No token provided' })

};

}

let decoded;

try {

decoded = jwt.verify(token, jwtSecret);

console.log('JWT token verified:', decoded);

} catch (err) {

console.error('Error verifying JWT token:', err);

return {

statusCode: 401,

headers: responseHeaders,

body: JSON.stringify({ message: 'Unauthorized: Invalid token' })

};

}

let body;

try {

body = JSON.parse(event.body);

} catch (err) {

console.error('Error parsing request body:', err);

return {

statusCode: 400,

headers: responseHeaders,

body: JSON.stringify({ message: 'Bad Request: Invalid JSON' })

};

}

const { imageType, image } = body;

if (!imageType || !image) {

return {

statusCode: 400,

headers: responseHeaders,

body: JSON.stringify({ message: 'Bad Request: imageType and image are required' })

};

}

// Determine folder based on imageType

const validImageTypes = ['profile', 'question', 'option', 'answer'];

if (!validImageTypes.includes(imageType)) {

return {

statusCode: 400,

headers: responseHeaders,

body: JSON.stringify({ message: `Bad Request: Invalid imageType. Valid types are: ${validImageTypes.join(', ')}` })

};

}

const folderName = imageType;

const key = `${folderName}/${Date.now()}\_image.jpg`; // Example key for saving in specified folder

const params = {

Bucket: bucketName,

Key: key,

Body: Buffer.from(image, 'base64'), // Assuming the image is base64 encoded

ContentType: 'image/jpeg' // Adjust accordingly if using a different image type

};

try {

const data = await s3.upload(params).promise();

console.log(`Image uploaded successfully to ${data.Location}`);

// Return response with image URL

return {

statusCode: 200,

headers: responseHeaders,

body: JSON.stringify({ message: 'Image uploaded successfully', imageUrl: data.Location })

};

} catch (err) {

console.error('Error uploading image:', err);

return {

statusCode: 500,

headers: responseHeaders,

body: JSON.stringify({ message: 'Internal Server Error: Failed to upload image' })

};

}

};