**Lambda Function Name: validateEmailN\_R**

**API Name: Module1\_Refactored**

**Resource Name: validateEmailN\_R**

**API URL:** [**https://aymk6mhnl3.execute-api.us-east-1.amazonaws.com/v1/userRegisterN\_R**](https://aymk6mhnl3.execute-api.us-east-1.amazonaws.com/v1/userRegisterN_R)

**Overview**

This API endpoint is designed to handle user registration by validating email addresses and storing user details in a MongoDB database. The endpoint is hosted on AWS Lambda and is part of the Module1\_Refactored API.

**Request**

**POST /v1/userRegisterN\_R**

* **Headers**:
  + Content-Type: application/json
* **Body Parameters**:
  + fullname (string, required): The full name of the user.
  + email (string, required): The email address of the user.
  + password (string, required): The password for the user account.
  + InstituteName (string, optional): The name of the user's institute.
* **Sample Request**:

json

{

"fullname": "John Doe",

"email": "john.doe@example.com",

"password": "password123",

"InstituteName": "Example Institute"

}

**Response**

**POST /v1/userRegisterN\_R**

* **Headers**:
  + Access-Control-Allow-Origin: \*
  + Access-Control-Allow-Headers: Content-Type
  + Access-Control-Allow-Credentials: true
  + Access-Control-Allow-Methods: GET,POST,OPTIONS
* **Response Status Codes**:
  + 201 Created: User registered successfully.
  + 400 Bad Request: Missing fields or invalid email format.
  + 400 Bad Request: Email already exists.
  + 500 Internal Server Error: Failed to register user.
* **Sample Responses**:
  + **Success**:

json

{

"message": "User registered successfully",

}

* + **Error** (Missing fields):

json

{

"message": "Missing fields"

}

* + **Error** (Invalid email format):

json

{

"message": "Invalid email format"

}

* + **Error** (Email already exists):

json

{

"message": "Email already exists"

}

* + **Error** (Internal server error):

json

{

"message": "Failed to register user"

}

CODE:

const mongoose = require('mongoose');

const User = require('./model.js');

const mongoUri = process.env.MONGO\_URI;

exports.handler = async (event) => {

let client;

const corsHeaders = {

'Access-Control-Allow-Origin': 'https://admin.exambuilder.online',

'Access-Control-Allow-Headers': 'Content-Type',

'Access-Control-Allow-Credentials': true,

'Access-Control-Allow-Methods': 'GET,POST,OPTIONS',

};

const responseHeaders = {

'Access-Control-Allow-Origin': 'https://admin.exambuilder.online',

};

// Handle preflight requests

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

return {

statusCode: 204,

headers: corsHeaders,

};

}

let payload;

try {

payload = JSON.parse(event.body);

} catch (error) {

return {

statusCode: 400,

headers: responseHeaders,

body: JSON.stringify({ message: 'Invalid JSON payload' }),

};

}

try {

client = await mongoose.connect(mongoUri, {

useNewUrlParser: true,

useUnifiedTopology: true,

});

const { fullname, email, password, InstituteName } = payload;

if (!fullname || !email || !password || !InstituteName) {

return {

statusCode: 400,

headers: responseHeaders,

body: JSON.stringify({ message: 'Missing fields' }),

};

}

const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

if (!emailRegex.test(email)) {

return {

statusCode: 400,

headers: responseHeaders,

body: JSON.stringify({ message: 'Invalid email format' }),

};

}

const existingEmail = await User.findOne({ email });

if (existingEmail) {

return {

statusCode: 409, // Conflict

headers: responseHeaders,

body: JSON.stringify({ message: 'Email already exists' }),

};

}

const user = new User({

fullname,

email,

InstituteName,

password,

});

await user.save();

return {

statusCode: 201,

headers: responseHeaders,

body: JSON.stringify({ message: 'User registered successfully' }),

};

} catch (error) {

console.error('Registration error:', error);

if (error instanceof mongoose.Error.ValidationError) {

return {

statusCode: 422, // Unprocessable Entity

headers: responseHeaders,

body: JSON.stringify({ message: 'Validation error', details: error.errors }),

};

}

return {

statusCode: 500,

headers: responseHeaders,

body: JSON.stringify({ message: 'Failed to register user' }),

};

} finally {

if (client) {

await mongoose.disconnect(); // Disconnect from MongoDB

}

}

};