**Lambda Function Name:** resetPasswordN\_R

**API Name:** Reset Password API

**Resource Name:** /v1/resetPasswordN\_R

**API URL:** <https://aymk6mhnl3.execute-api.us-east-1.amazonaws.com/v1/resetPasswordN_R>

**1. Role of the API**

The API allows users to reset their password by updating it in the database for a given email address.

**2. Functioning**

* **POST Request**: Accepts a JSON payload containing the email and newPassword.
  + Connects to MongoDB using the provided MONGO\_URL.
  + Finds the user by their email address (case-insensitive).
  + If the user is found, updates their password with the newPassword.
  + Responds with status code 200 if successful, 404 if the user is not found, and 500 if there's an internal server error.
* **OPTIONS Request (Preflight)**: Handles CORS by responding with allowed methods (OPTIONS, POST) and headers (Content-Type).

**3. API Payload**

* **POST Payload:**

json

{

"email": "user@example.com",

"newPassword": "newPassword123"

}

* + **email**: The user's email address.
  + **newPassword**: The new password to be set for the user.

**4. Response**

* **Successful Response (200):**

json

{

"message": "Password updated successfully"

}

* **User Not Found (404):**

json

{

"message": "User not found"

}

* **Internal Server Error (500):**

json

{

"message": "Internal server error"

}

**5. Dependency**

* **MongoDB**: The API depends on MongoDB (MONGO\_URL environment variable) to store and retrieve user data.

**6. Logic**

* Parses the incoming request body to extract email and newPassword.
* Connects to MongoDB to find the user by email.
* Updates the user's password if found.
* Handles CORS preflight OPTIONS request to allow cross-origin requests.
* Responds with appropriate status codes and messages based on the success or failure of the operation.

CODE:

const mongoose = require('mongoose');

const User = require('./User'); // Assuming you have a User model

exports.handler = async (event) => {

const corsHeaders = {

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

};

const preflightHeaders = {

...corsHeaders,

'Access-Control-Allow-Methods': 'OPTIONS, POST', // Allow only OPTIONS and POST methods

'Access-Control-Allow-Headers': 'Content-Type' // Allow Content-Type header

};

// Handle preflight OPTIONS request

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

return {

statusCode: 200,

headers: preflightHeaders,

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

};

}

let payload;

try {

payload = JSON.parse(event.body);

} catch (jsonError) {

console.error('Invalid JSON payload:', jsonError);

return {

statusCode: 400, // Bad request

headers: corsHeaders,

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

};

}

const { email, newPassword } = payload;

try {

// Connect to MongoDB using environment variable for MongoDB URI

await mongoose.connect(process.env.MONGO\_URL, { useNewUrlParser: true, useUnifiedTopology: true });

const lowercaseEmail = email.toLowerCase();

// Find the user by email

const user = await User.findOne({ email: lowercaseEmail });

if (!user) {

mongoose.disconnect(); // Disconnect from MongoDB

return {

statusCode: 404, // User not found

headers: corsHeaders,

body: JSON.stringify({ message: 'User not found' })

};

}

// Update user's password with the hashed password received from the frontend

user.password = newPassword;

await user.save();

mongoose.disconnect(); // Disconnect from MongoDB

return {

statusCode: 200, // Password updated successfully

headers: corsHeaders,

body: JSON.stringify({ message: 'Password updated successfully' })

};

} catch (error) {

console.error('Error updating password:', error);

mongoose.disconnect(); // Disconnect from MongoDB in case of error

return {

statusCode: 500, // Internal server error

headers: corsHeaders,

body: JSON.stringify({ message: 'Internal server error' })

};

}

};