PROJECT

SERVERLESS WEB APPLICATION

STEP1: CREATE A REST API OF NAME GetEmployeeDetailsByEmail



Graphical user interface, text, application, email

Description automatically generated

Step2:Create a resource in the new API and name it as GetCustomerDetailsByEmail.Make sure you enable the CORS while creation of resource.

Graphical user interface

Description automatically generated

A screenshot of a computer

Description automatically generated

Step 3: Create A GET method in the resource.

A screenshot of a computer

Description automatically generated

Step 4:Until the creation of lambda set the integration type to mock for GET method.

Graphical user interface, text, application

Description automatically generated

Step 5:Go to integration Response and expand mapping templates.

A screenshot of a computer

Description automatically generated

a)it is set to application/Json

b)select it and set the compiler to error

c)paste the desired output you want.

{

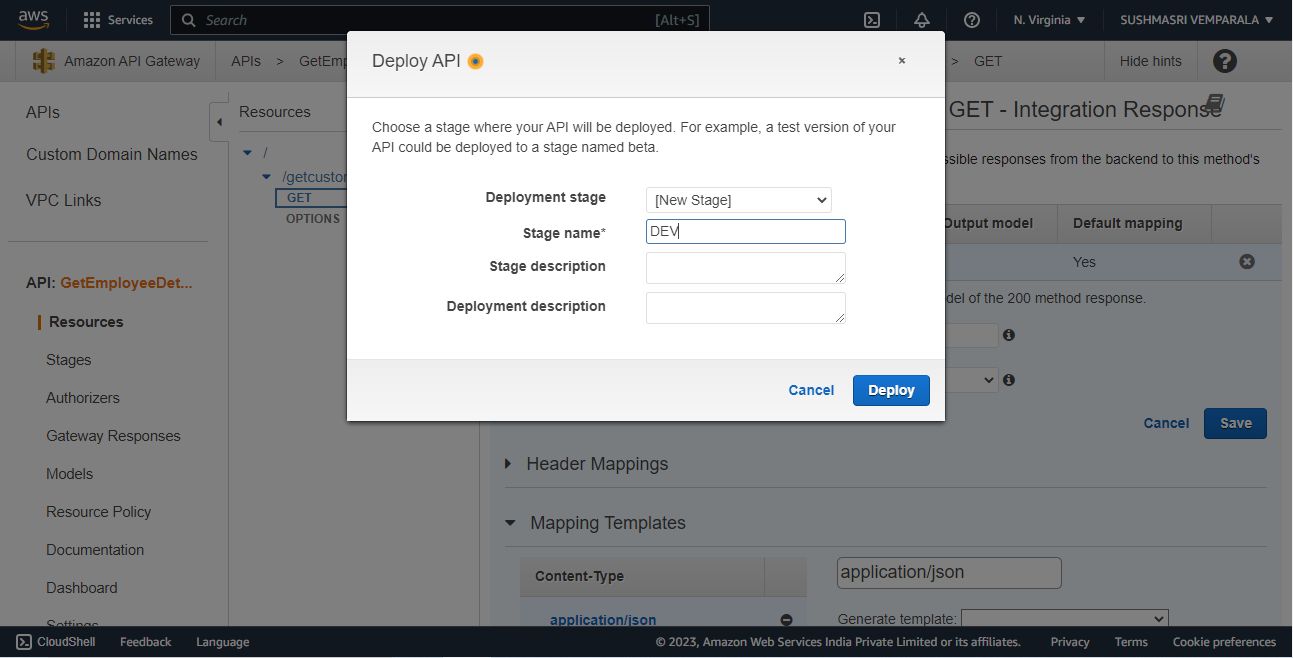
"FirstName":"sushmasri",

"LastName":"Vemparala"

"Email":"sushmasrivemparala24@gmail.com"

}

STEP 6: After saving the changes You need to deploy the changes of API by creating new Stage.



Step7: After deploying an API you will get a Invoke url. Check this Invoke URL in postman whether it is working or not.

Graphical user interface, text, email

Description automatically generated

At first you will get missing authentication error. This is because it root url of api so add HTTP methos created at the end of invoke url.

<https://xa3m58ykj7.execute-api.us-east-1.amazonaws.com/DEV/getcustomerdetailsbyemail>

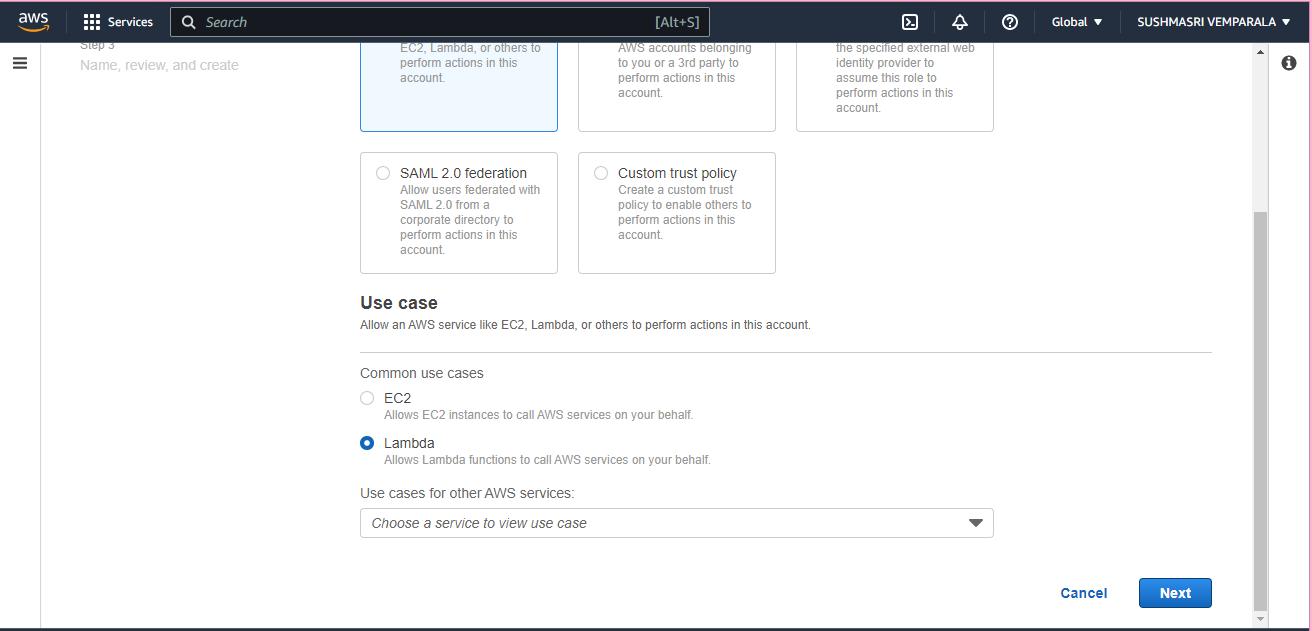
Graphical user interface, text, application, email

Description automatically generated

So postman can retrieve our data using url successfully. API is working properly.

**STEP 8**: NOW ITS TIME IMPLEMENT BUSINESS LOGIC USING LAMBDA.IN ORDER TO CREATE FUNCTION WE NEED TO DEFINE A ROLE WHICH HAS LAMBDA

$AWSLambdaFullAcess $ AWSDynamoDBReadAccess



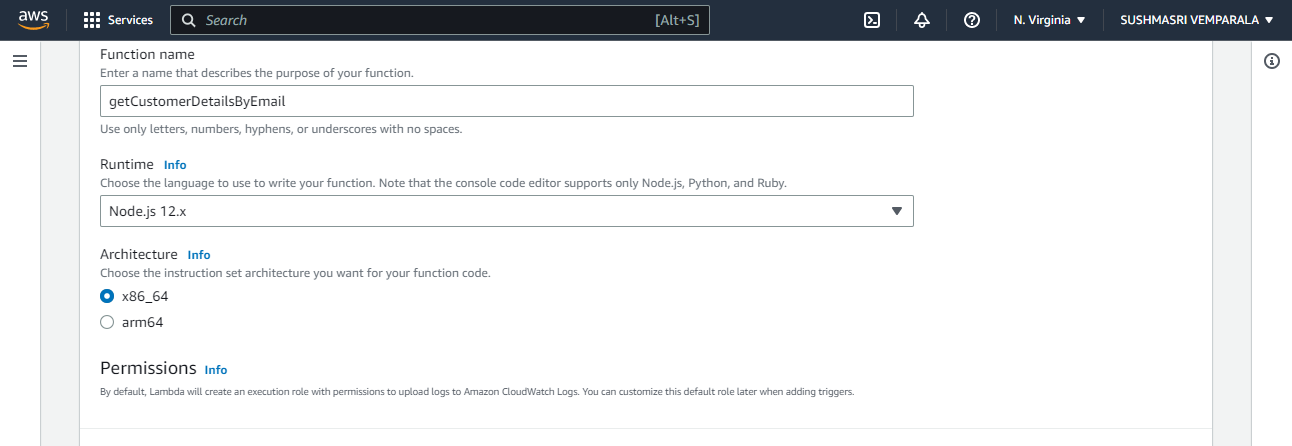
Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

STEP 9: NOW CREATE LAMBDA FUCNTION AND ATTACH ROLE TO IT.



Graphical user interface, text, application, email

Description automatically generated

exports.handler = async (event) => {

// TODO implement

const response = {

statusCode: 200,

body: JSON.stringify('Hello from Lambda!'),

};

return response;

};

Graphical user interface, text, application

Description automatically generated

Give details like that and test the code.

Graphical user interface, text, application

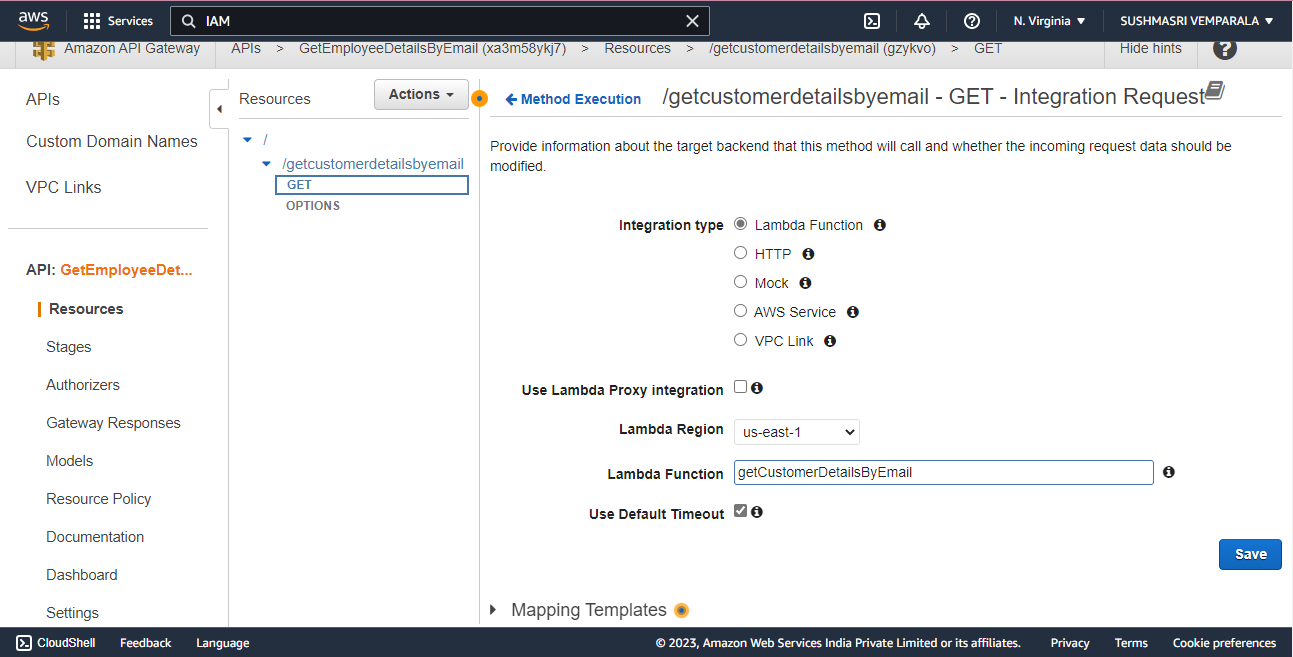
Description automatically generated

You will get like that.

STEP:10 : IT’s time to integrate API with lambda by converting integration request from mock to lambda.

Graphical user interface, text, application

Description automatically generated



Graphical user interface, text, application

Description automatically generated

AGAIN DEPLOY API OF DEV STAGE

STEP 11: CHECK INTEGRATION IN POSTMAN . WHEN YOU HIT URL WE SHOULD GET RESULT FROM LAMBDA

Graphical user interface, text, application, email

Description automatically generated

STEP 12: NOW CREATE DYNAMODB AND SET EMAILID AS PARITIONKEY.AND CREATE ITEMS.

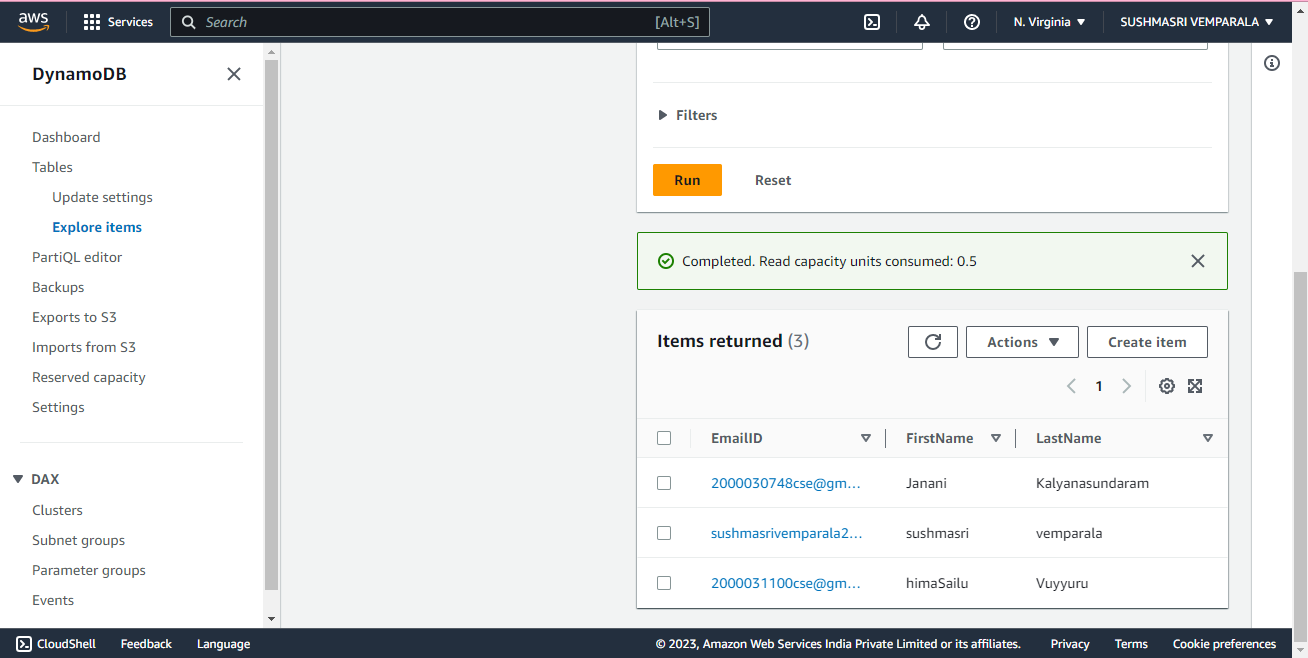
Graphical user interface, text, website

Description automatically generated

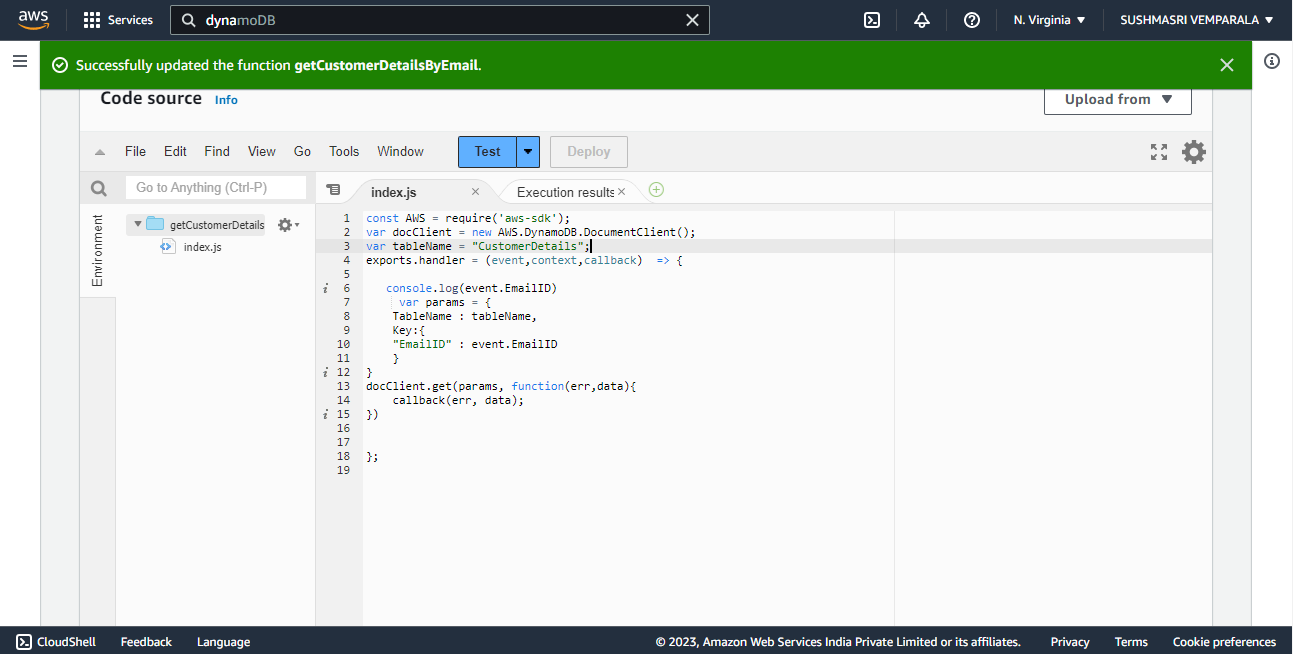
Graphical user interface, text, email

Description automatically generated

Similarly create some more records/items.



STEP 13: WRITE LAMBDA CODE THAT RETRIEVES THE INFORMATION FROM DYNAMODB



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

var docClient = new AWS.DynamoDB.DocumentClient();

var tableName = "CustomerDetails";

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

console.log(event.EmailID)

var params = {

TableName : tableName,

Key:{

"EmailID" : event.EmailID

}

}

docClient.get(params, function(err,data){

callback(err, data);

})

};

while testing the code give your email in test event of hello-world template.

In output it should retrieve your information from the dynamodb.

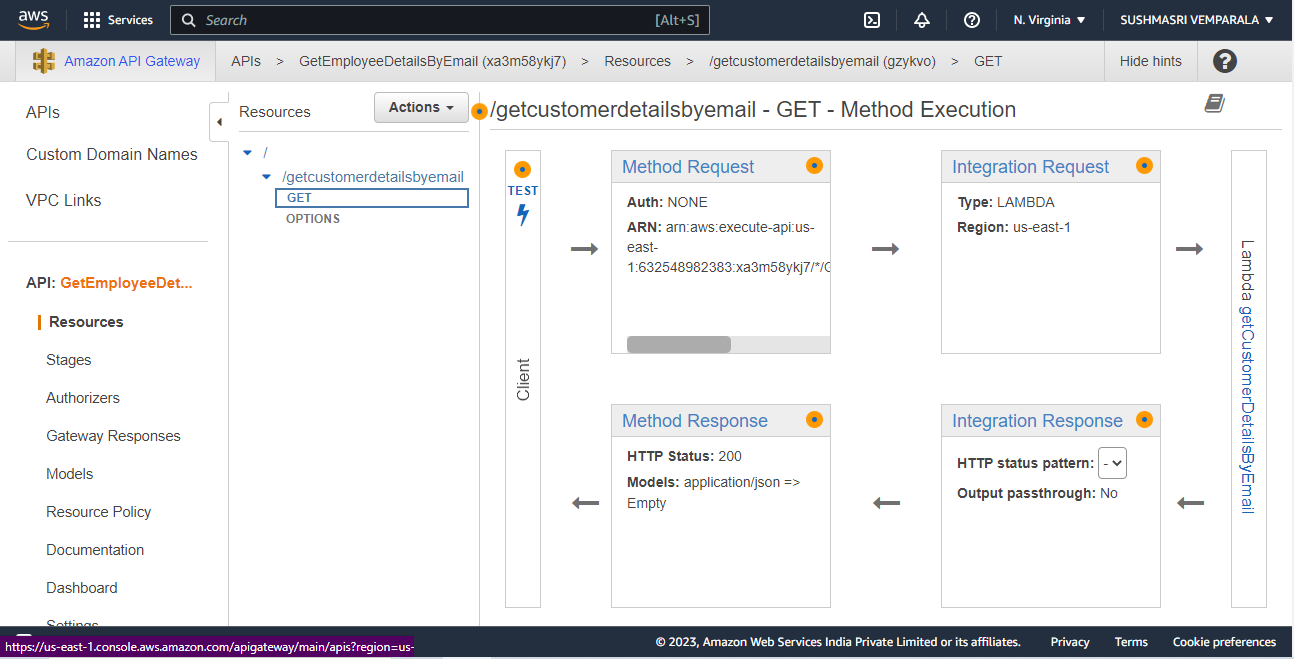
Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

STEP 14: NOW GO TO API ADD METHOD REQUEST TO API. QUERY STRING IS EMAIL.



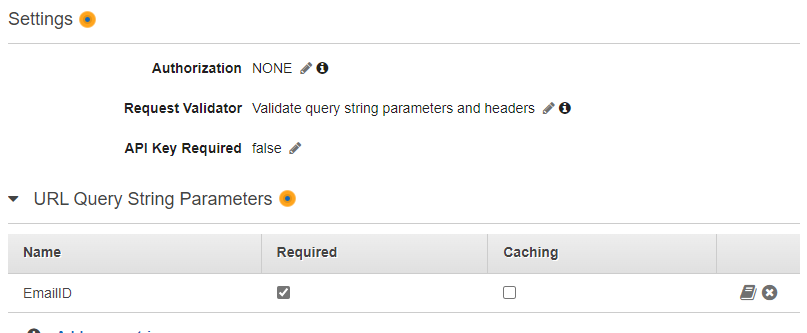
ADD QUERY STRING AS EMAILID

A screenshot of a computer

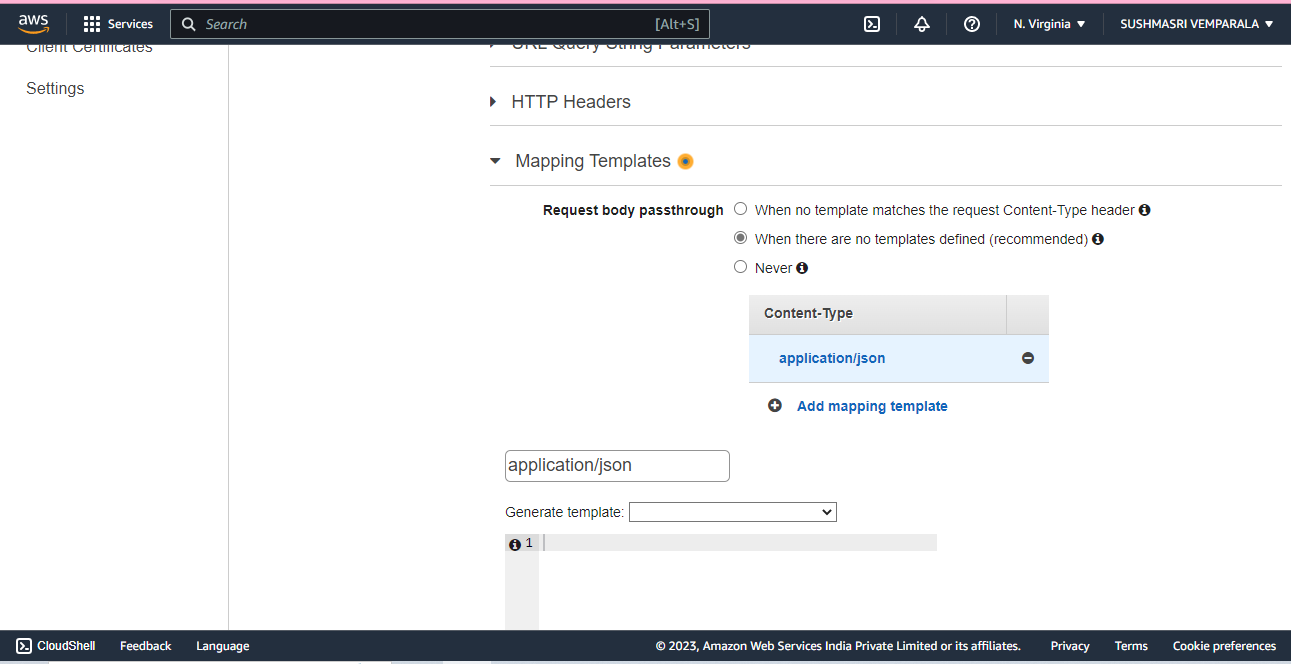
Description automatically generated

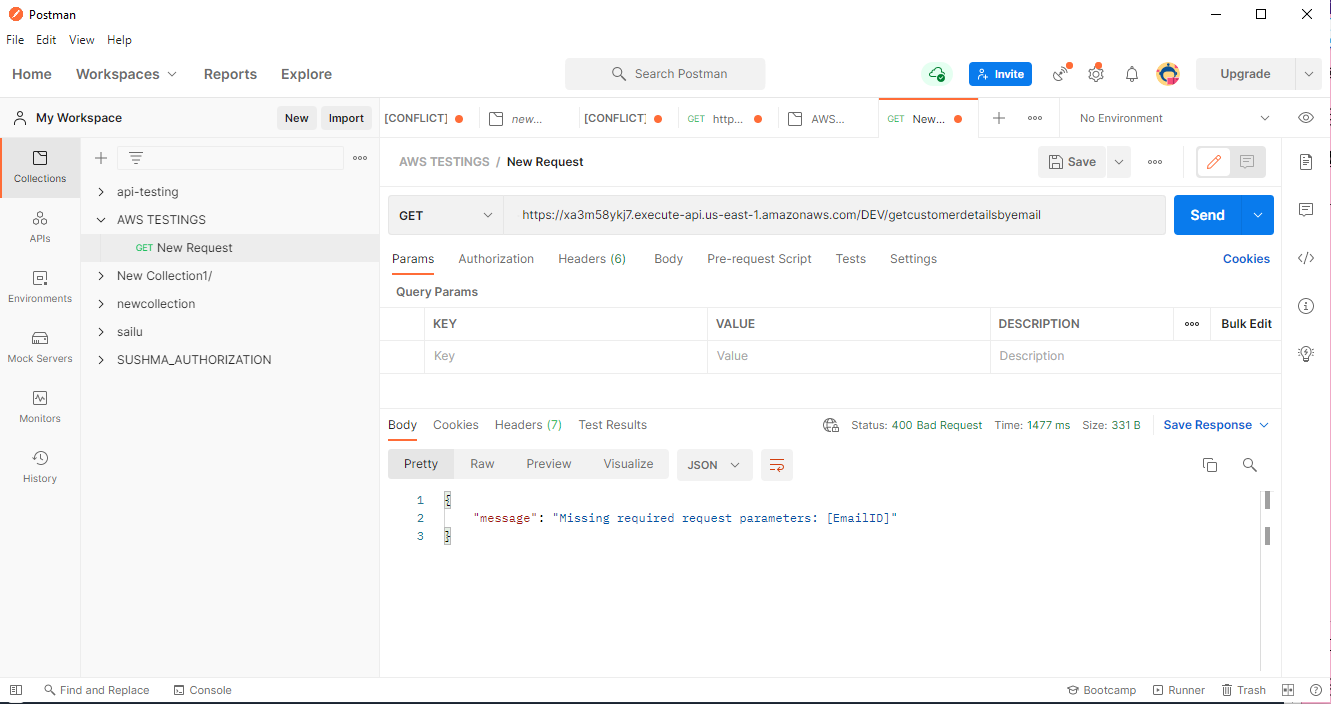
Graphical user interface, text, application, chat or text message

Description automatically generated

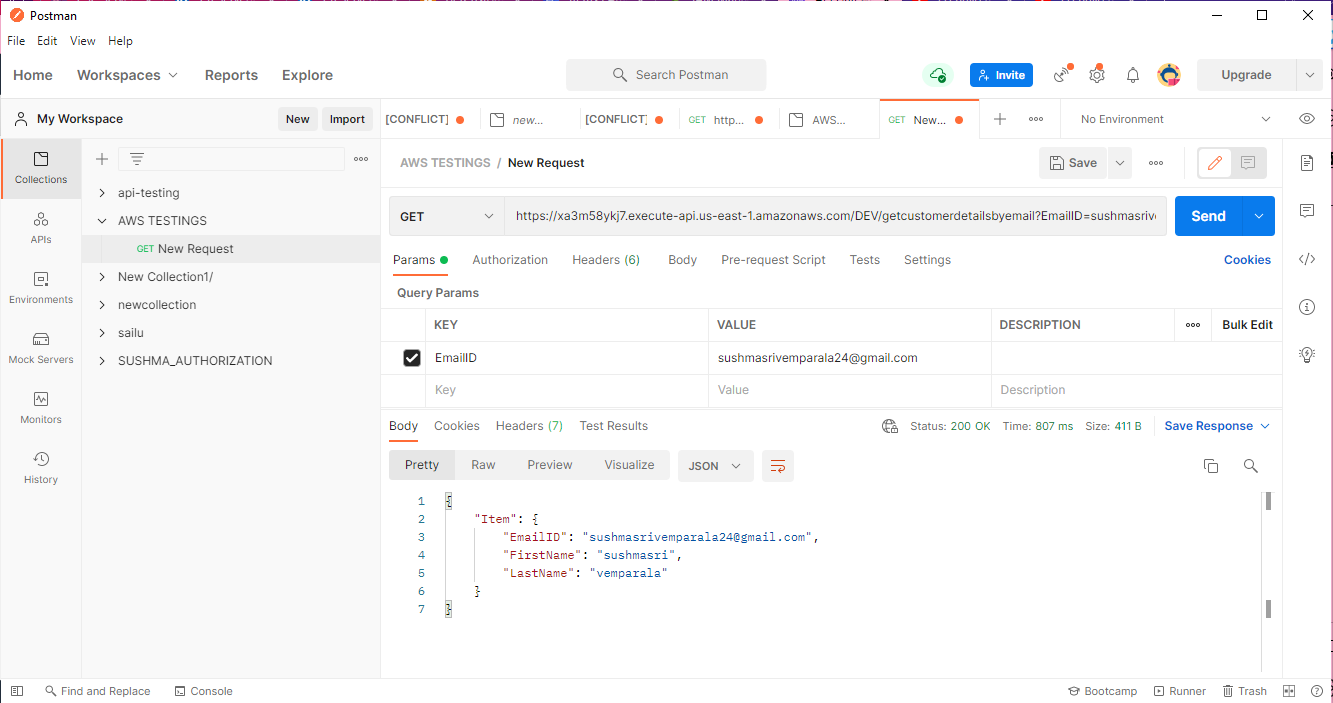


STEP 15: NOW GO TO INTEGRATION REQUEST AND SPECIFY EMAIL AS INPUT PARAMS.





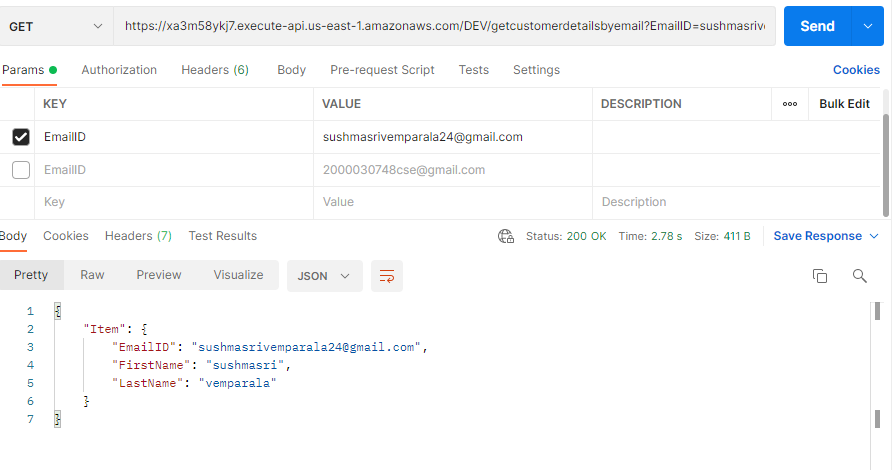
GIVE EMAIL AS KEY VALUES .



Graphical user interface, text, application, email

Description automatically generated

Now we have to get the output in browser for that we need s3.



Text

Description automatically generated

Paste your api url here.

MAKE SURE YOU ENABLE CORS.

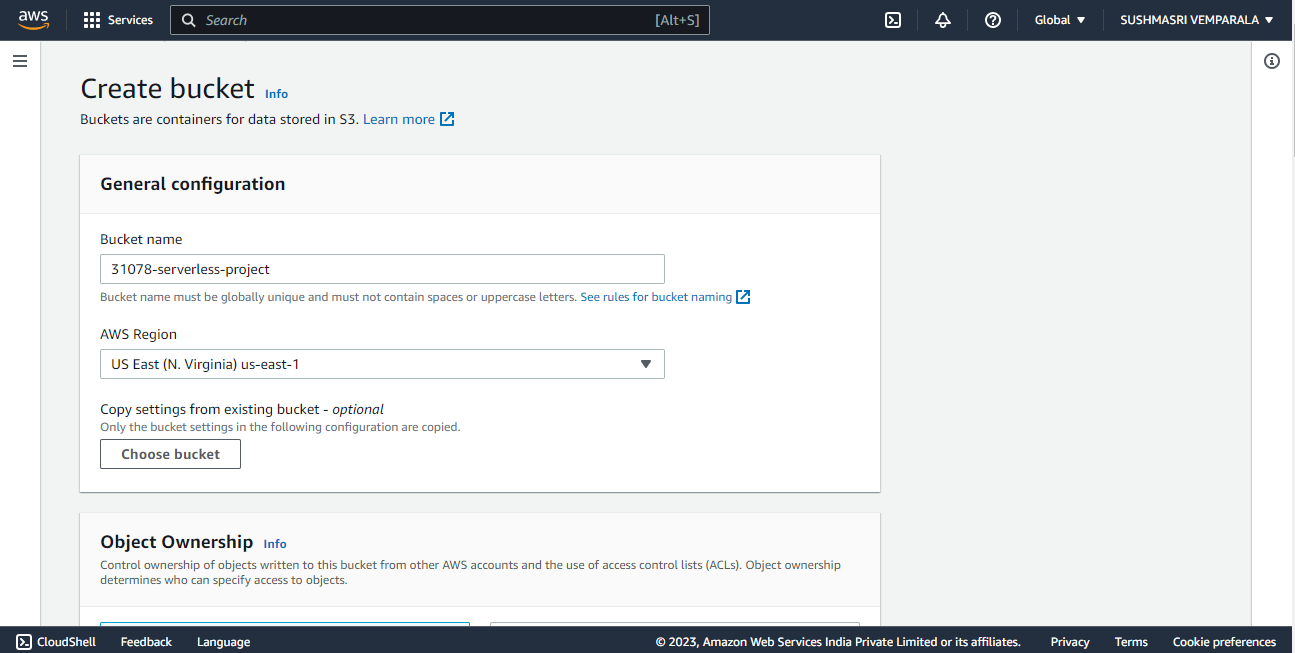
Graphical user interface, text, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

STEP 16: NOW CREATE THE S3 BUCKET AND PUT THE FRONTEND FILES IN IT.

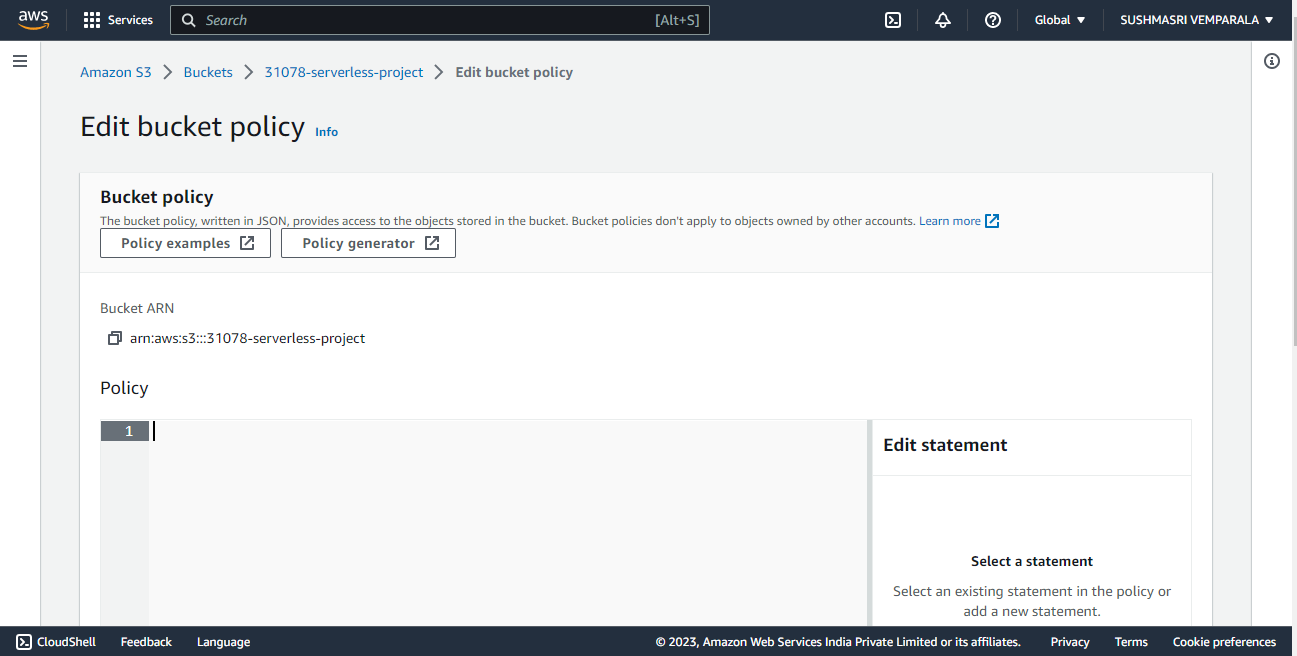


Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated



Graphical user interface, text, application, email

Description automatically generated

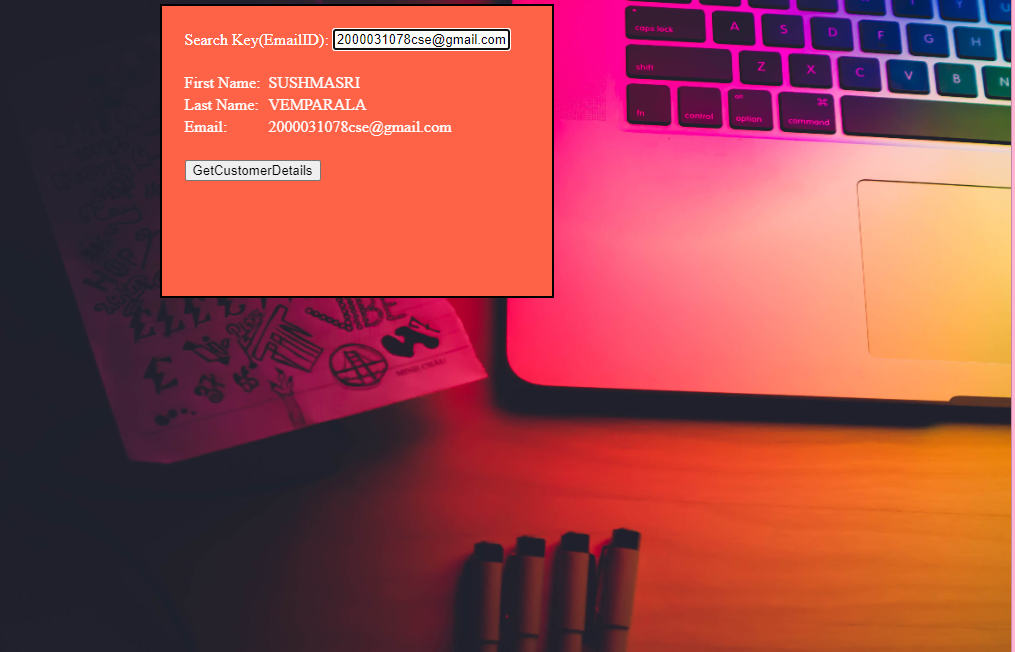
Graphical user interface, text, application

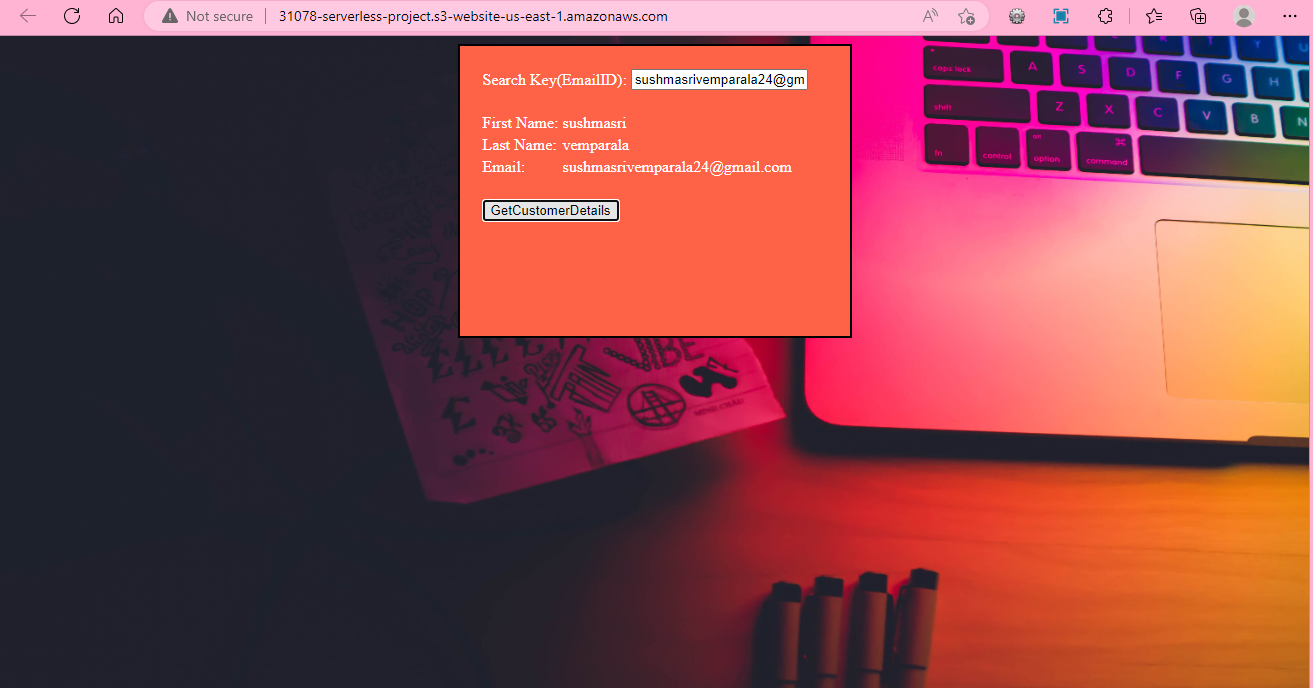
Description automatically generated

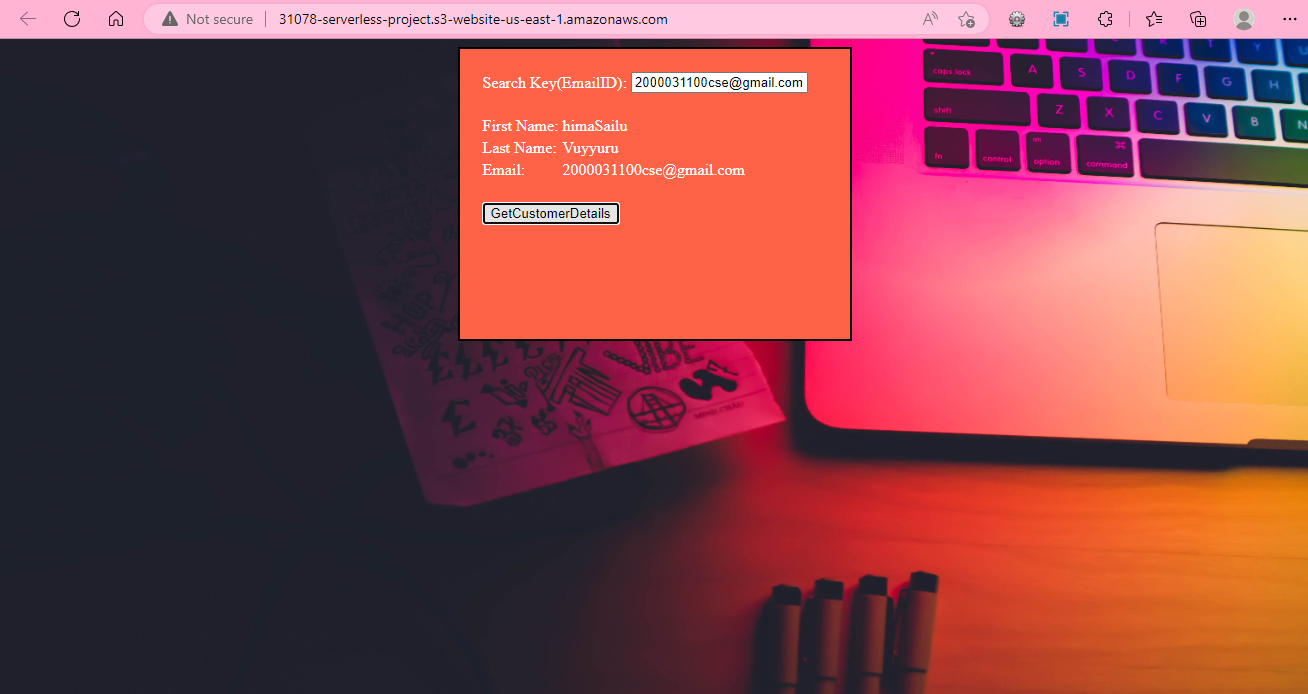
Graphical user interface, application, Word

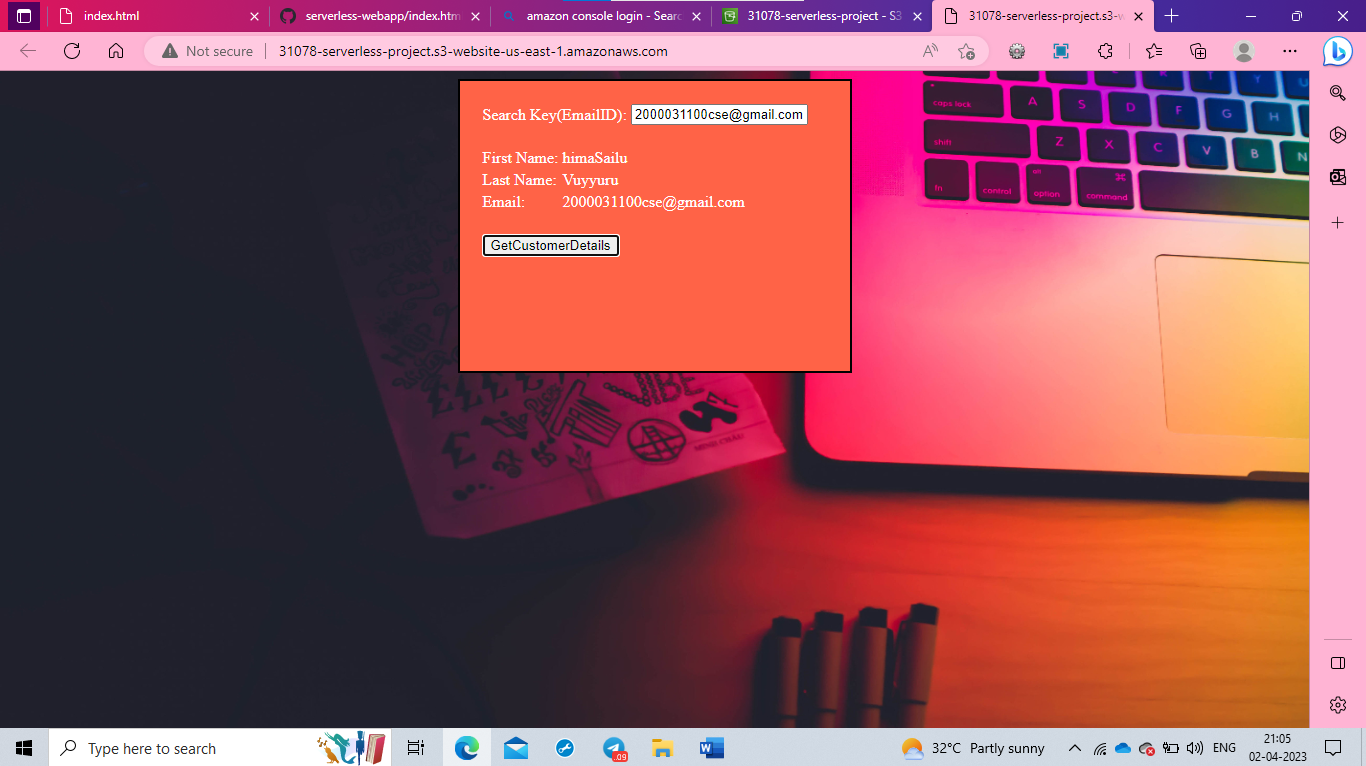
Description automatically generated

WE CAN CUTOMIZE THIS HTML PAGE BY ADDING BACKGROUND IMAGES AND DIV TAGS.







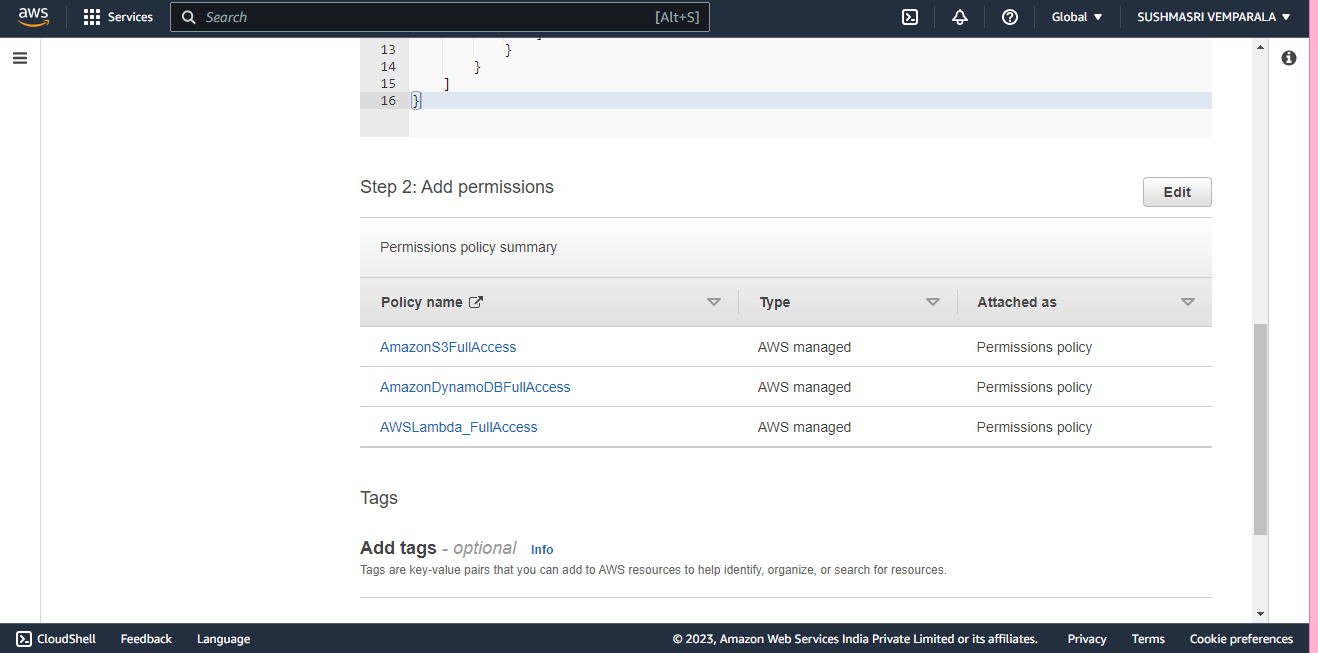


PART-2:

STEP 17: NOW BUILD THE POST APPLICATION.

FIRST CREATE A ROLE WITH 3 POLICIES.

[AmazonDynamoDBReadOnlyAccess](https://us-east-1.console.aws.amazon.com/iam/home#/policies/arn:aws:iam::aws:policy/AmazonDynamoDBReadOnlyAccess), [AmazonS3FullAccess](https://us-east-1.console.aws.amazon.com/iam/home#/policies/arn:aws:iam::aws:policy/AmazonS3FullAccess), [AWSLambda\_FullAccess](https://us-east-1.console.aws.amazon.com/iam/home#/policies/arn:aws:iam::aws:policy/AWSLambda_FullAccess)



Step 18: Now create lambda with above created role. This time give python 3.9 as runtime.



import json

import boto3

from time import gmtime, strftime

dynamodb = boto3.resource('dynamodb')

table = dynamodb.Table('Register')

now = strftime("%a, %d %b %Y %H:%M:%S +0000", gmtime())

def lambda\_handler(event, context):

# write name and time to the DynamoDB table using the object we instantiated and save response in a variable

    response = table.put\_item(

        Item={

            'Email': event['Email'],

            'FirstName': event['FirstName'],

            'LastName': event['LastName'],

            'EventType': event['EventType'],

            'EventName': event['EventName],

            })

# return a properly formatted JSON object

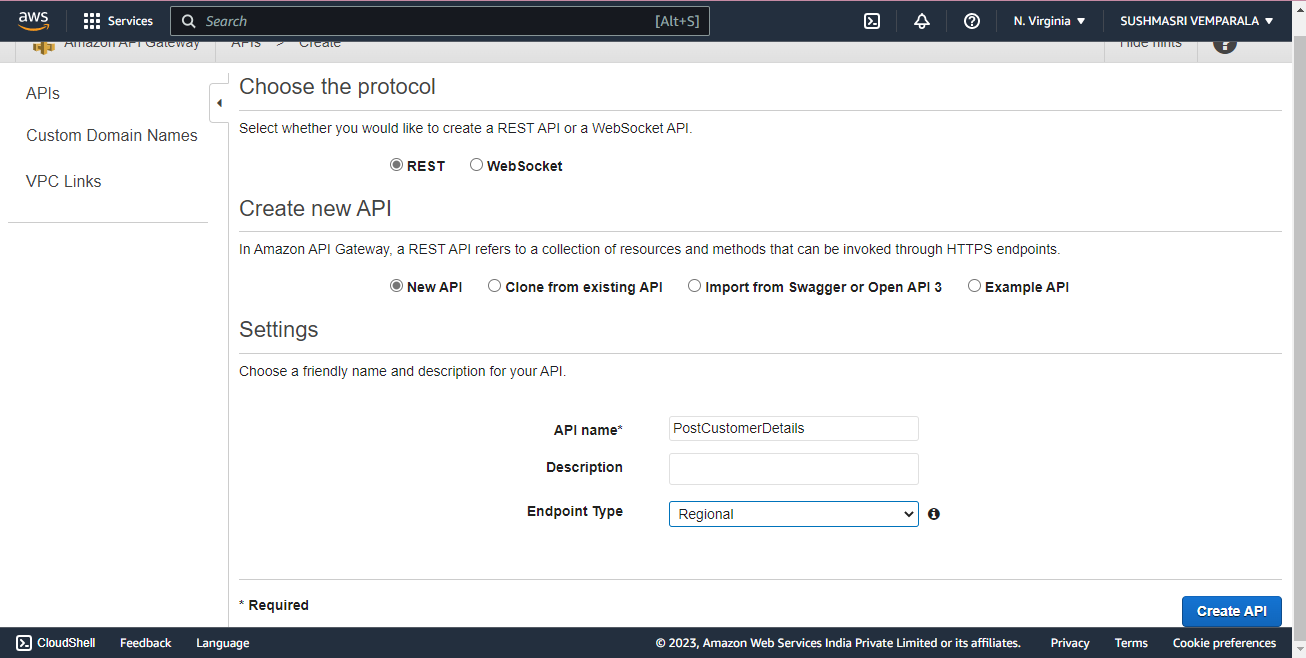
    return {

        'statusCode': 200,

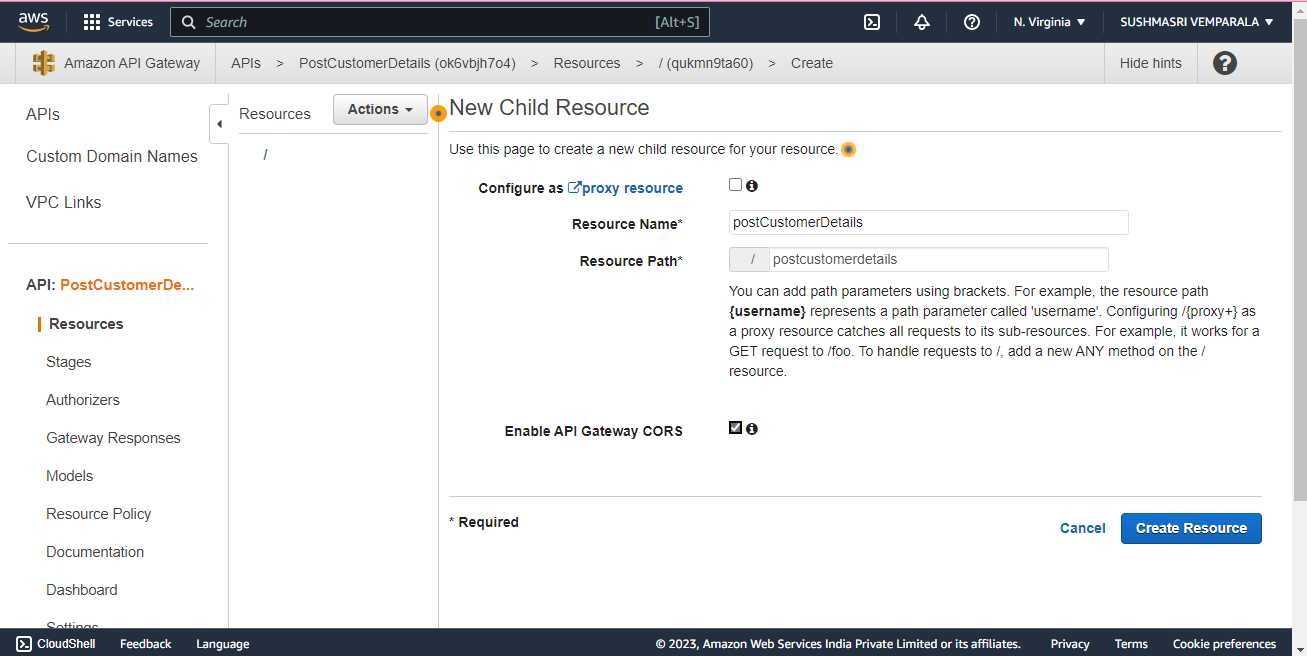
        'body': json.dumps('Hello from Lambda')

    }

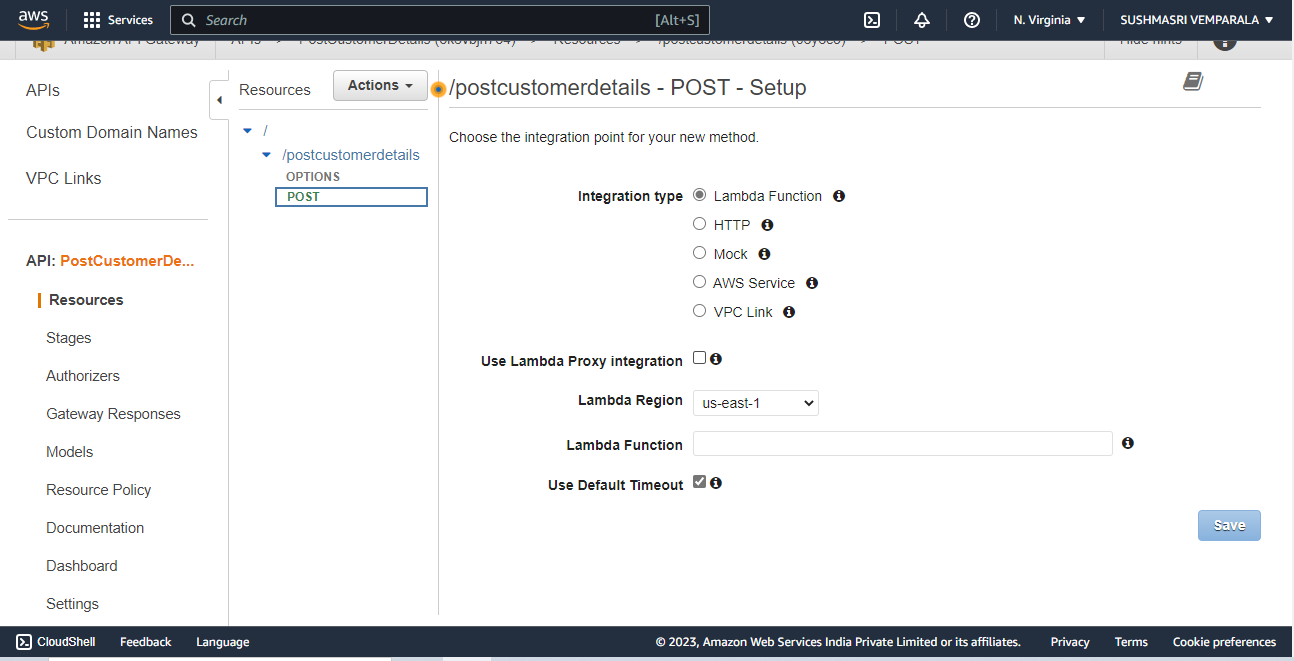
STEP 19: NOW CREATE API FOR POST DETAILS



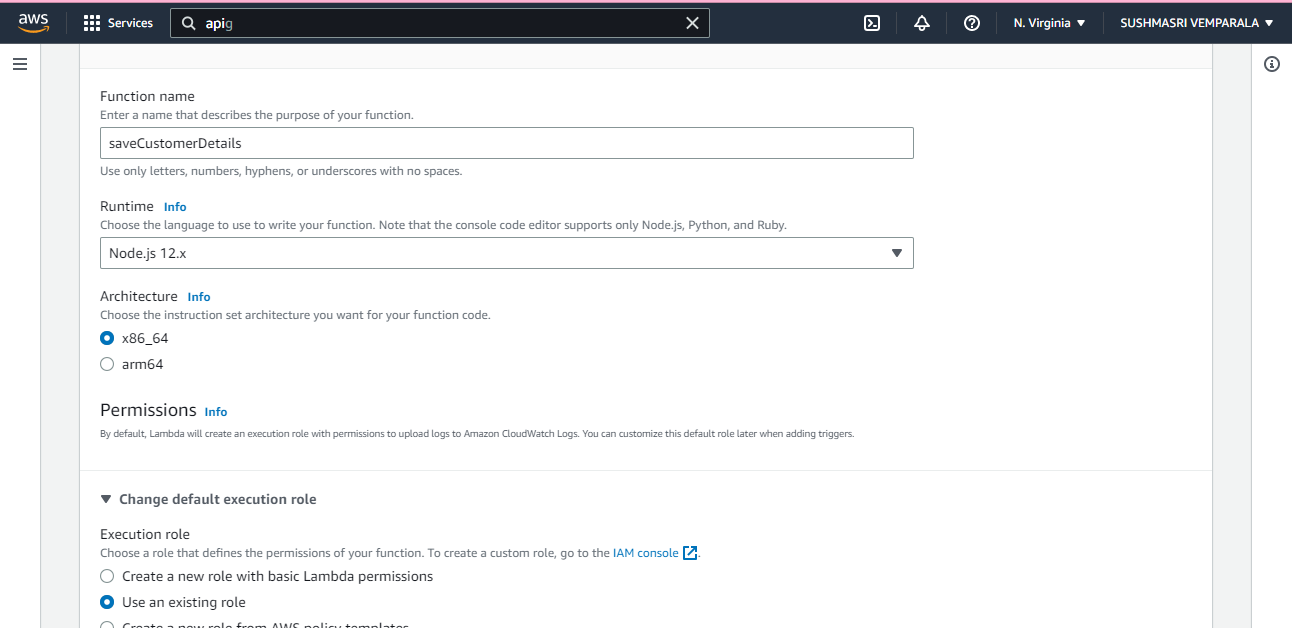
Create resources and enable the cors.



Create a POST method.



Step 20:go to s3 create .json function and attach the role.



Write the code in json

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

var docClient = new AWS.DynamoDB.DocumentClient();

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

var tableName = "CustomerDetails";

var params = {

TableName : tableName,

Item : {

"EmailID": event.EmailID,

"FirstName": event.FirstName,

"LastName":event.LastName

}

};

docClient.put(params,function(err,data){

if(err){

callback(err)

}

else

{

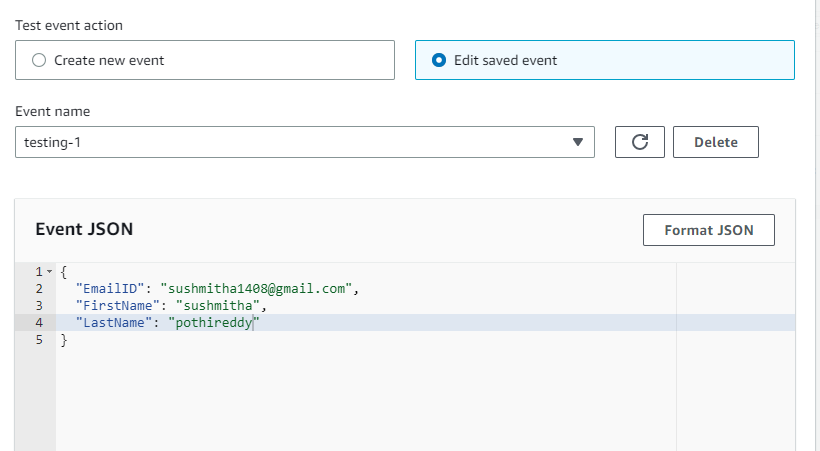
callback(null,"Successfully updated data")

}

})

};

Configure the test event like this



Test the lambda code function.

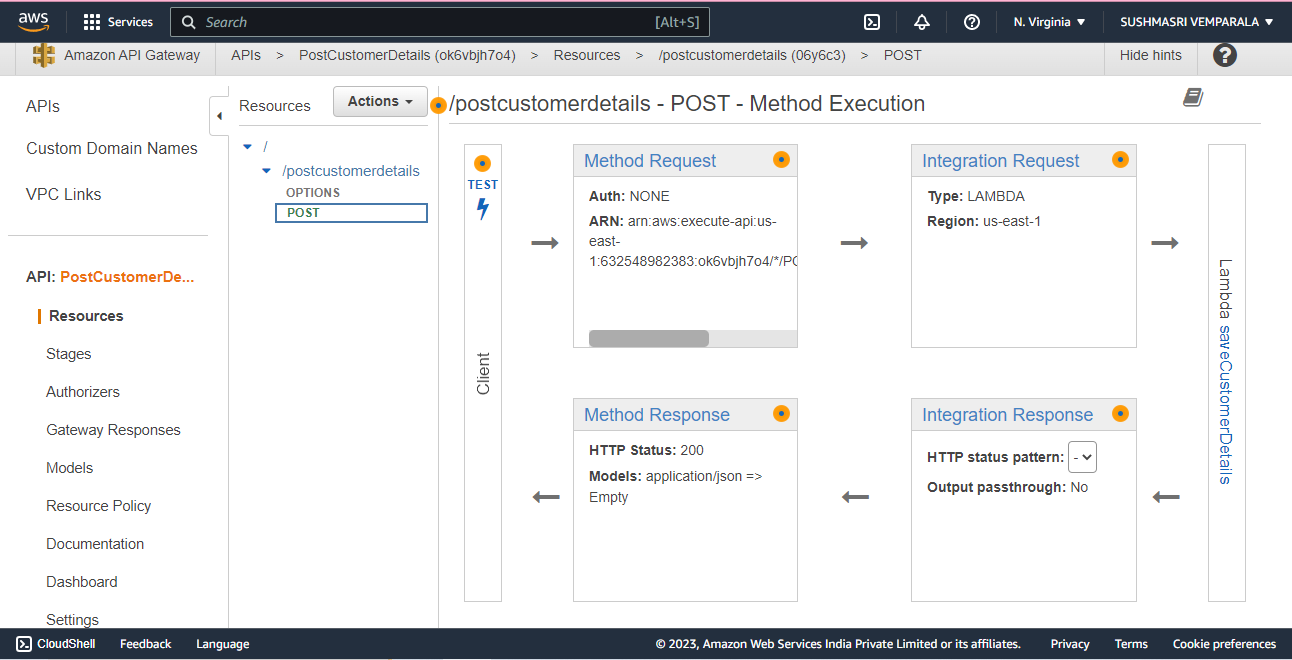
Graphical user interface, text, application

Description automatically generated



The data got added to database.

STEP 21: NOW ADD THE LAMBDA FUNCTION TO API GATEWAY.



ADD MAPPING TEMPLATE

Graphical user interface, text, application, email

Description automatically generated

Save and enable cors and deploy it.

DEPLOY API AND CHECK IT IN POST MAN

Graphical user interface, text, application, email

Description automatically generated

Go to body->raw->json and paste the configure event of lambda function. O/p successfully updated

FROM API GATEWAY IT IS ADDED TO DYNAMODB.

STEP 22: NOW CREATE S3 BUCKET AND ADD THE FRONT END FILES

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated



Graphical user interface, text, application, email

Description automatically generated

AFTER SAVING YOU WILL GET LIKE THAT

Graphical user interface, text, application

Description automatically generated

THEN GO TO DYNAOMDB AND SEARCH FOR ASHWIN RECORD

Graphical user interface, text, application, email, Teams

Description automatically generated