CSCI 5410: Assignment 3

Part A: Build an event-driven serverless application using AWS Lambda

GitLab Repositorhy Link: https://git.cs.dal.ca/drshah/dhrumilrakeshshah_csci5410.git

Screenshots of the Part A:

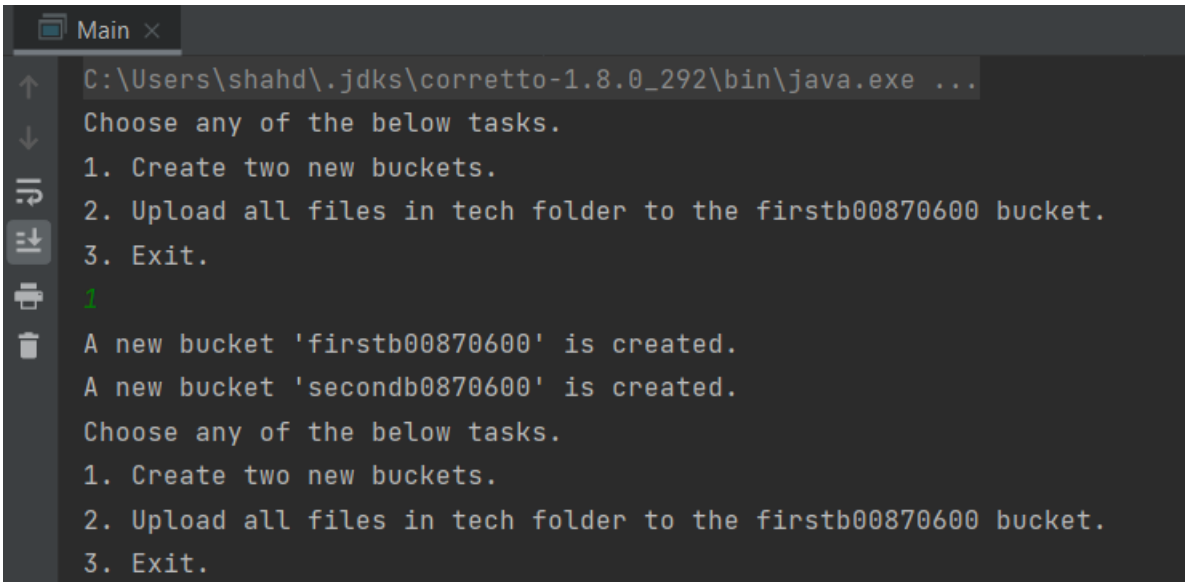

Figure 1: Two S3 Buckets Created 1

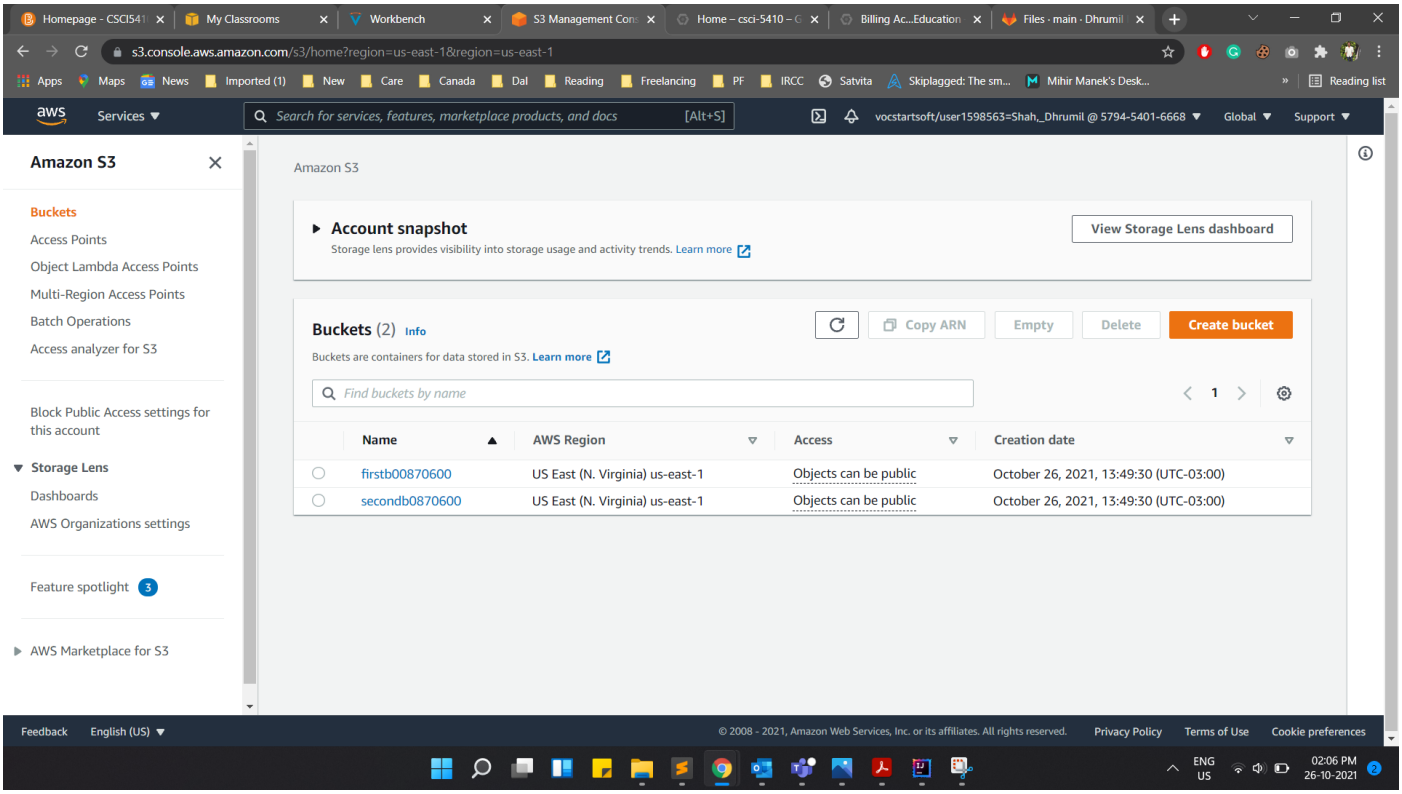

Figure 2: Two S3 Buckets Created 2

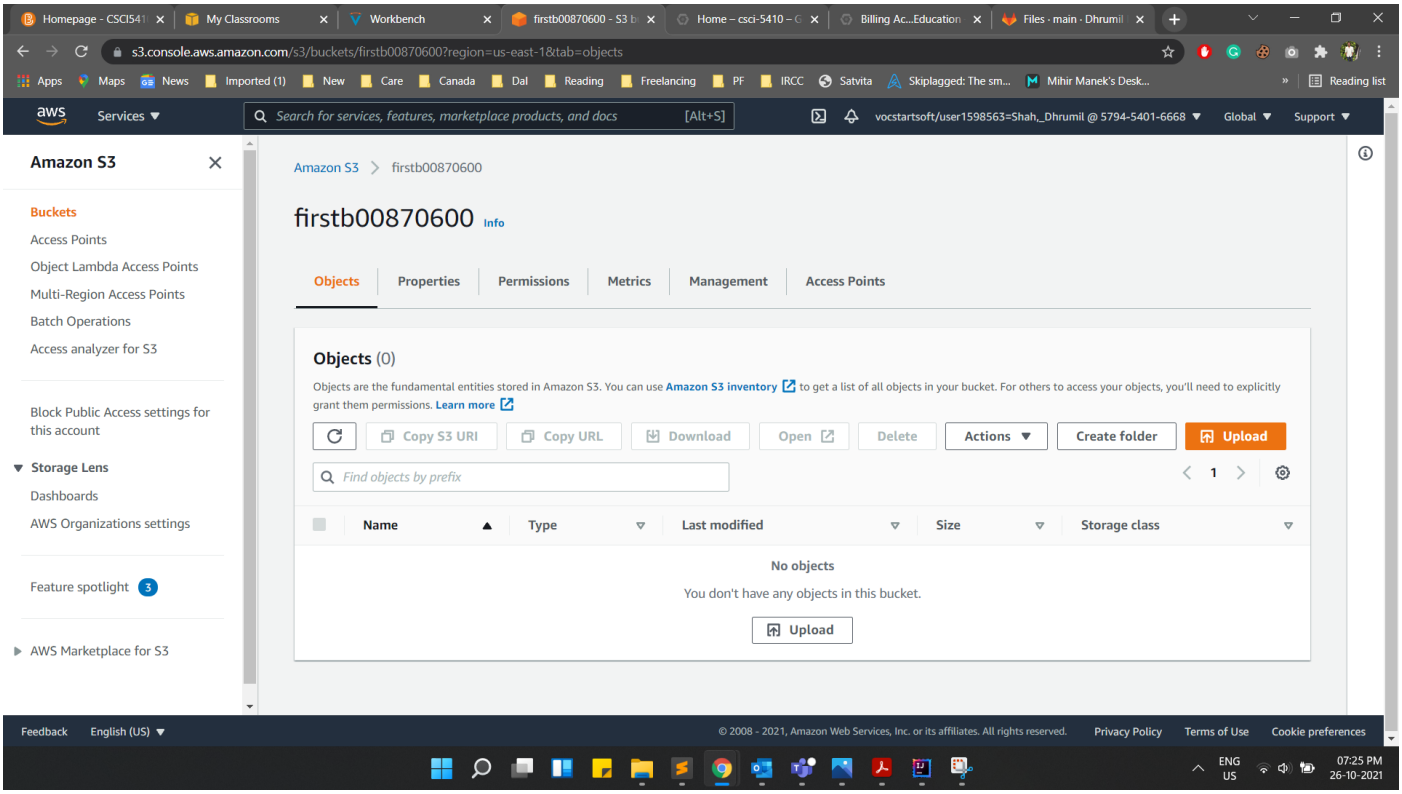

Figure 3: First Empty Bucket (firstb00870600)

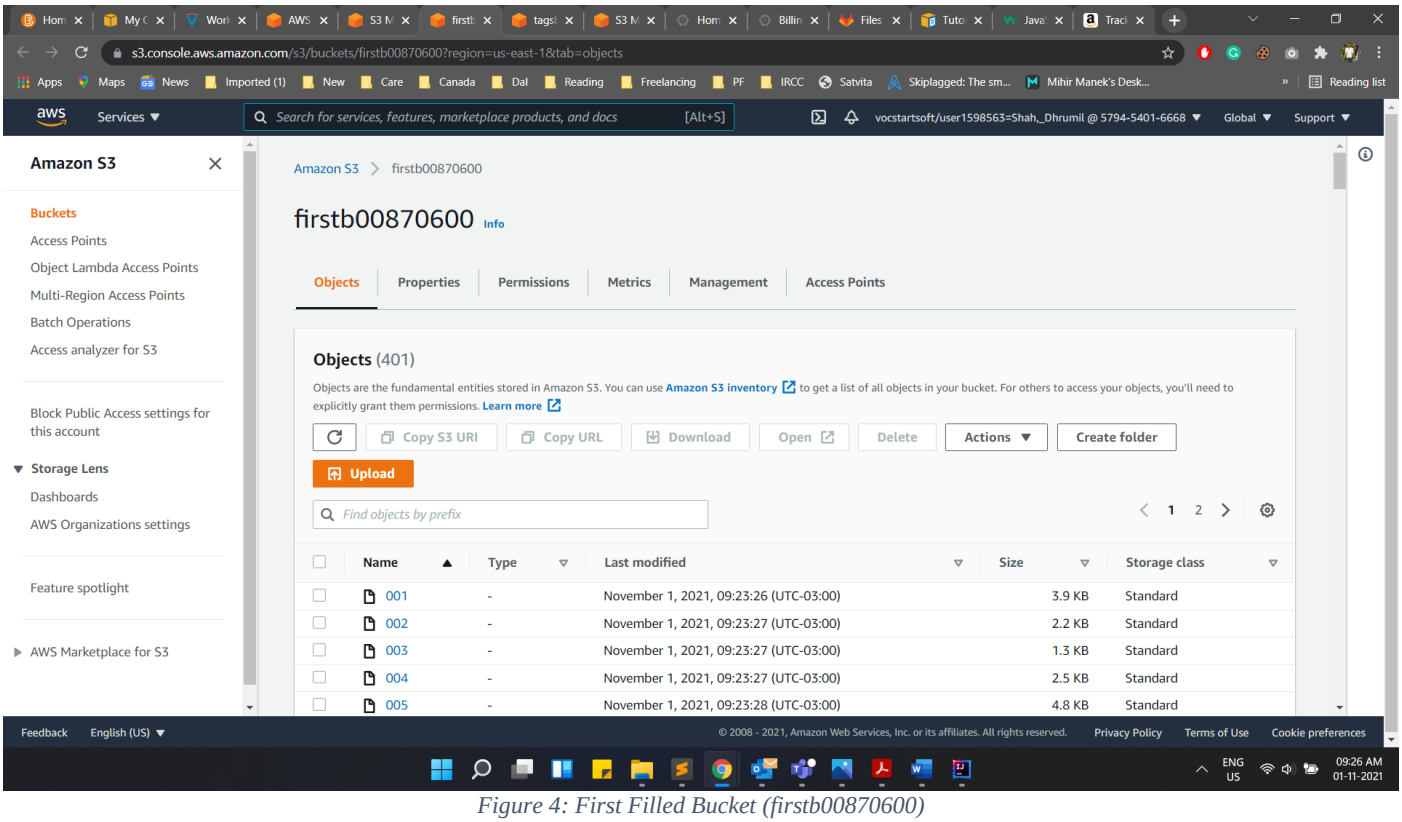

Figure 4: First Filled Bucket (firstb00870600)

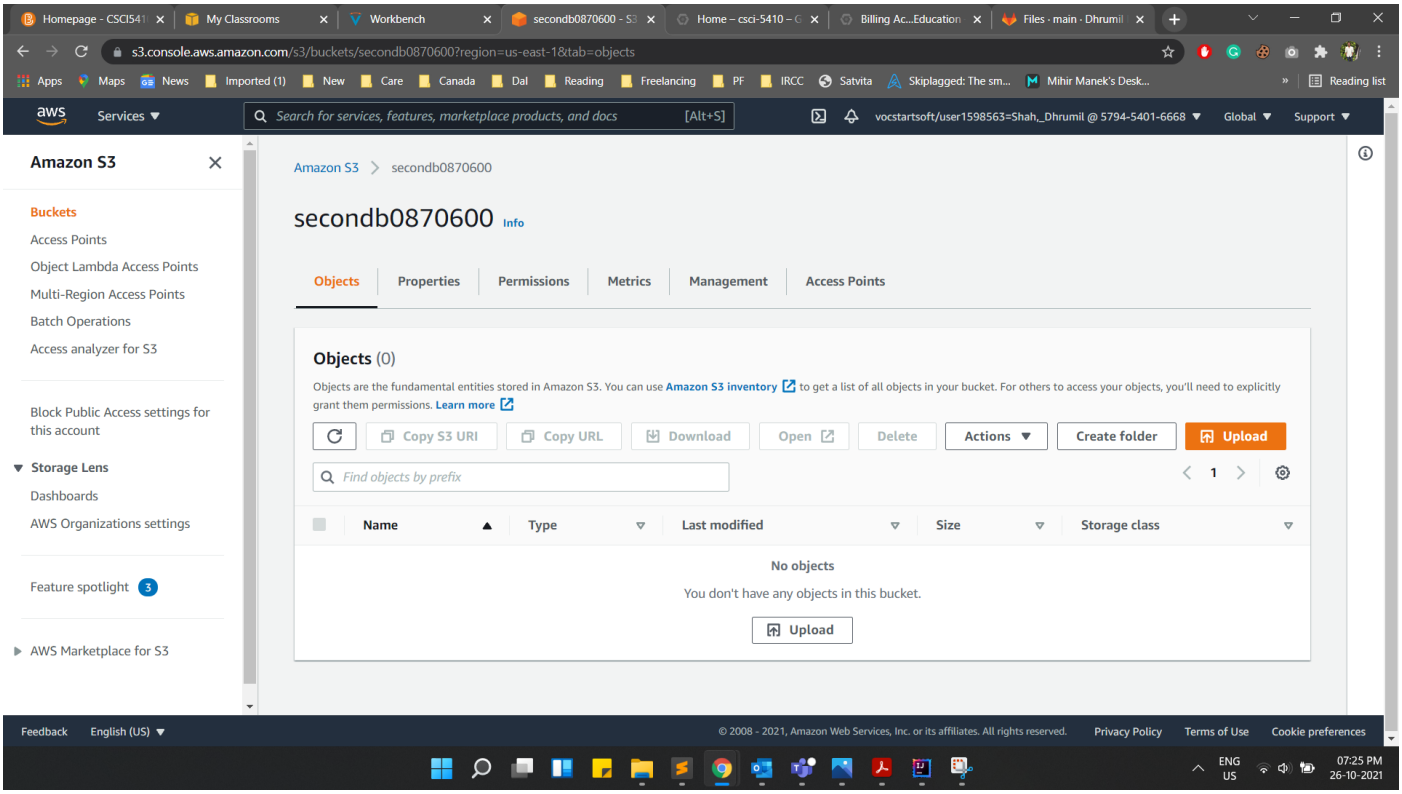

Figure 5: Second Empty Bucket (secondb00870600)

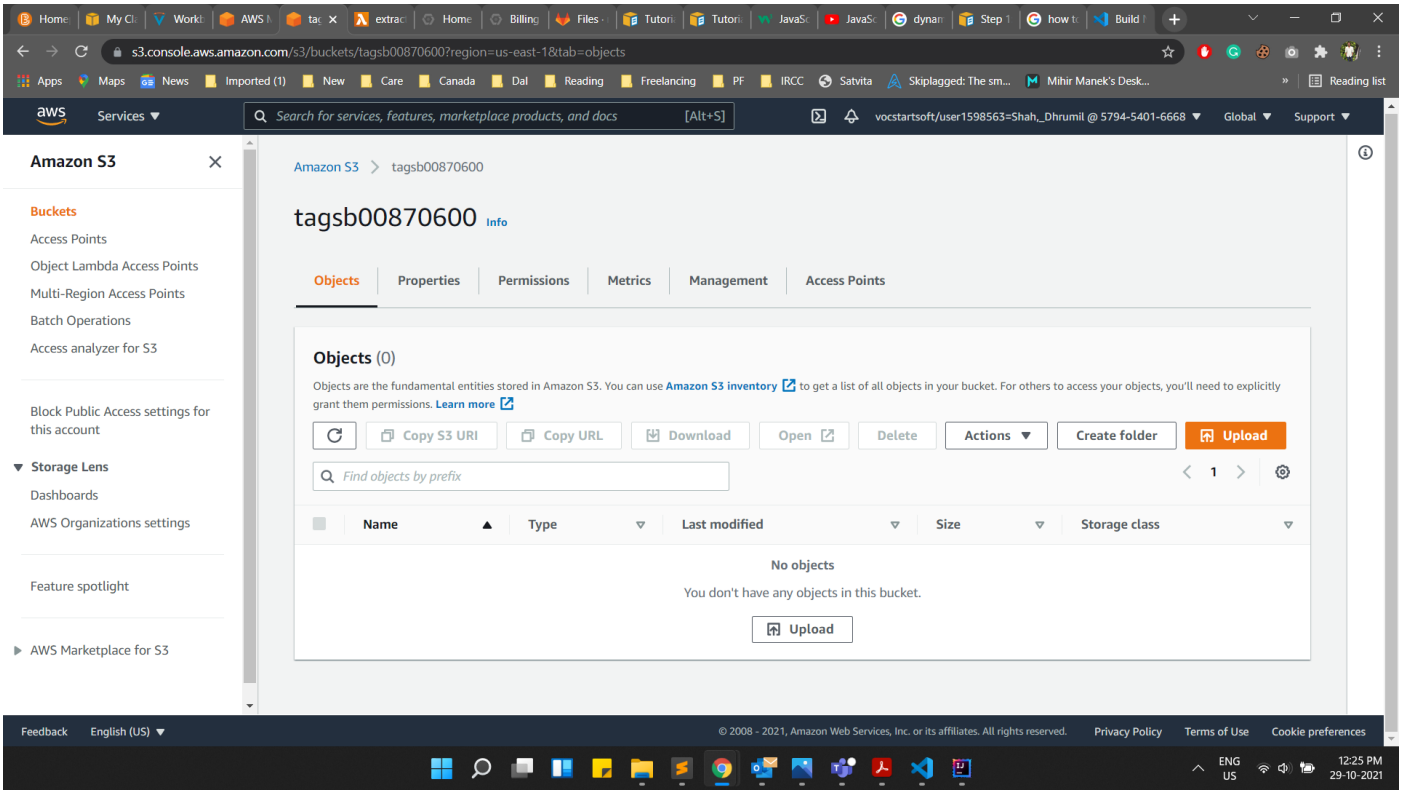

Figure 6: Third Empty Tags Bucket (tagsb00870600)

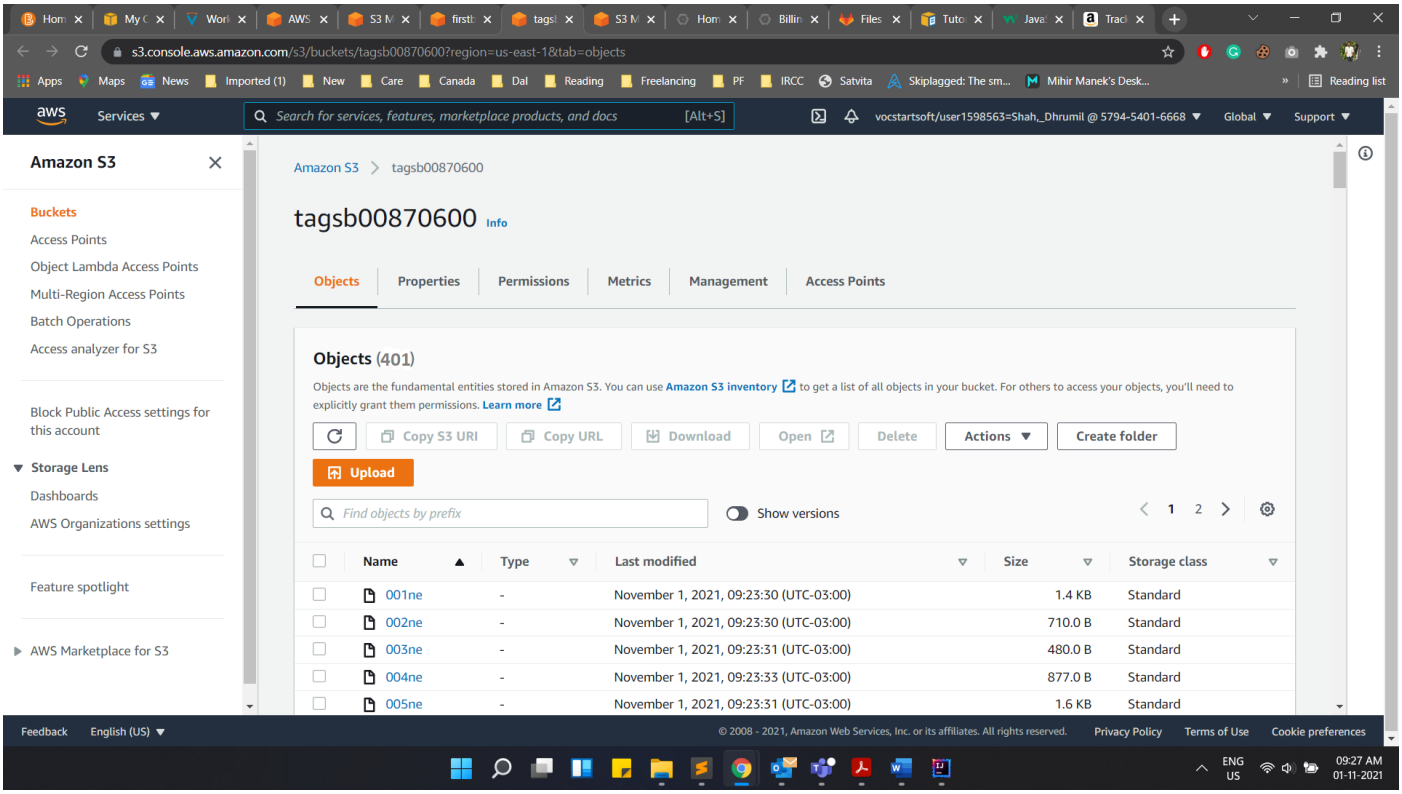

Figure 7: Third Filled Tags Bucket (tagsb00870600)

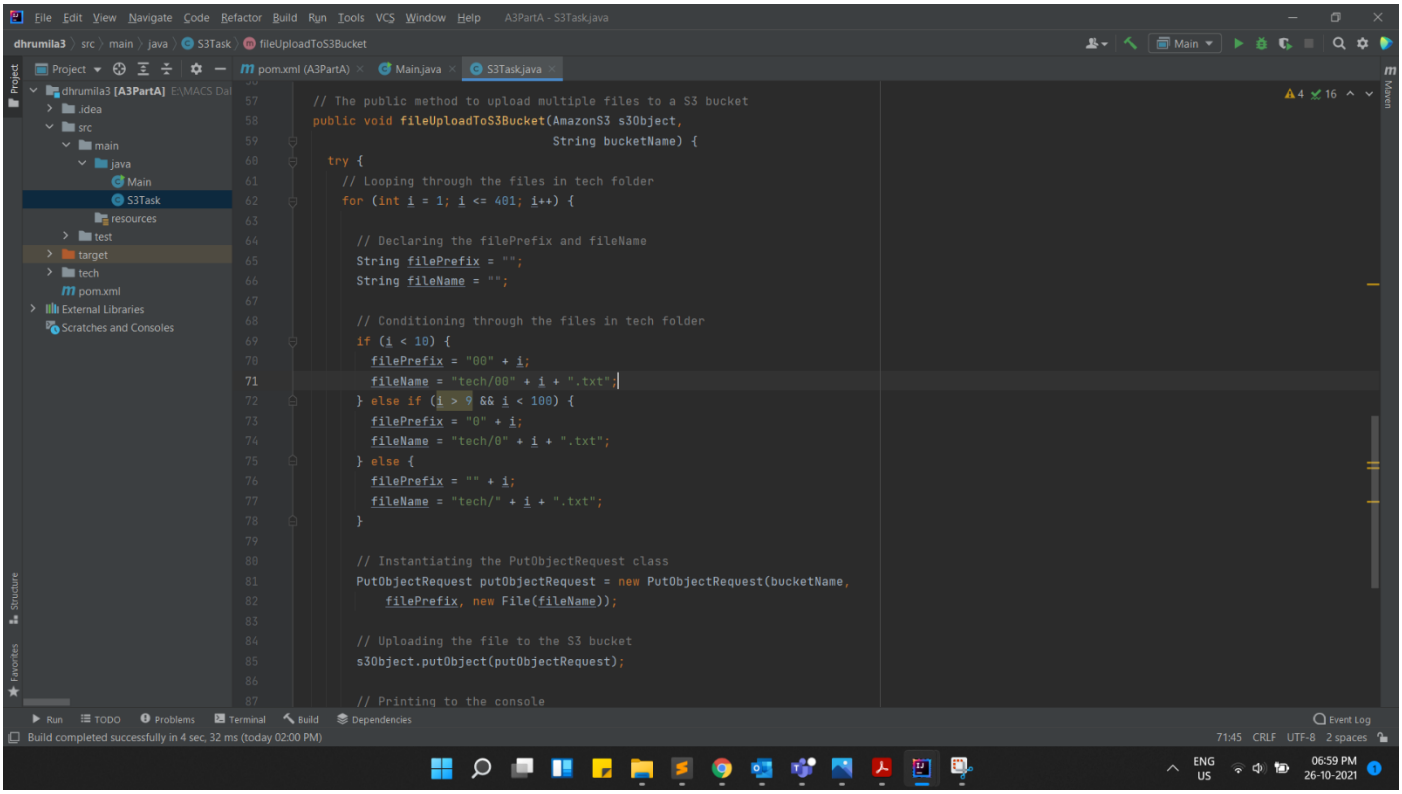

Figure 8: S3Task Code to Upload the Files to the S3 Bucket 1

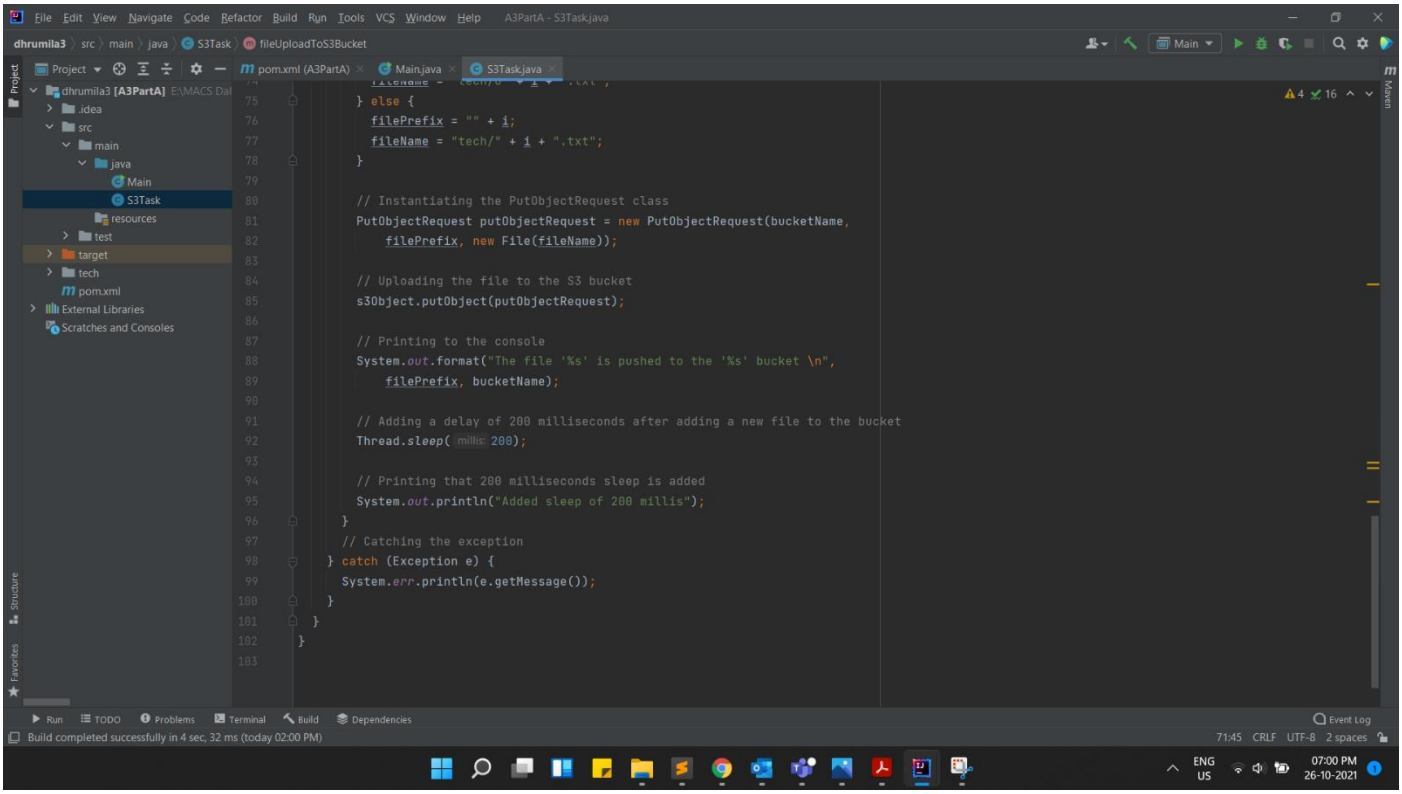

Figure 9: S3Task Code to Upload the Files to the S3 Bucket 2

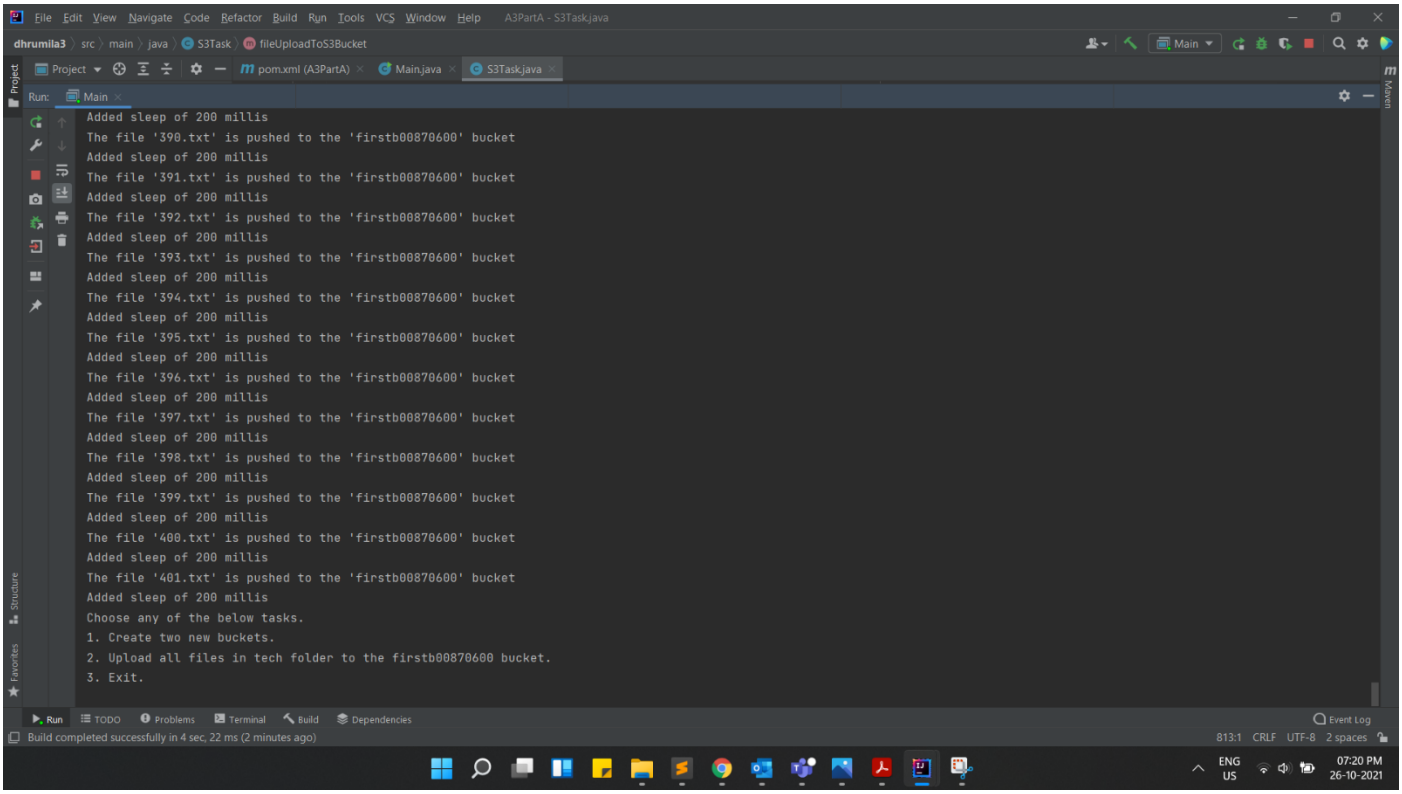

Figure 10: S3Task Code to Upload the Files to the S3 Bucket 3

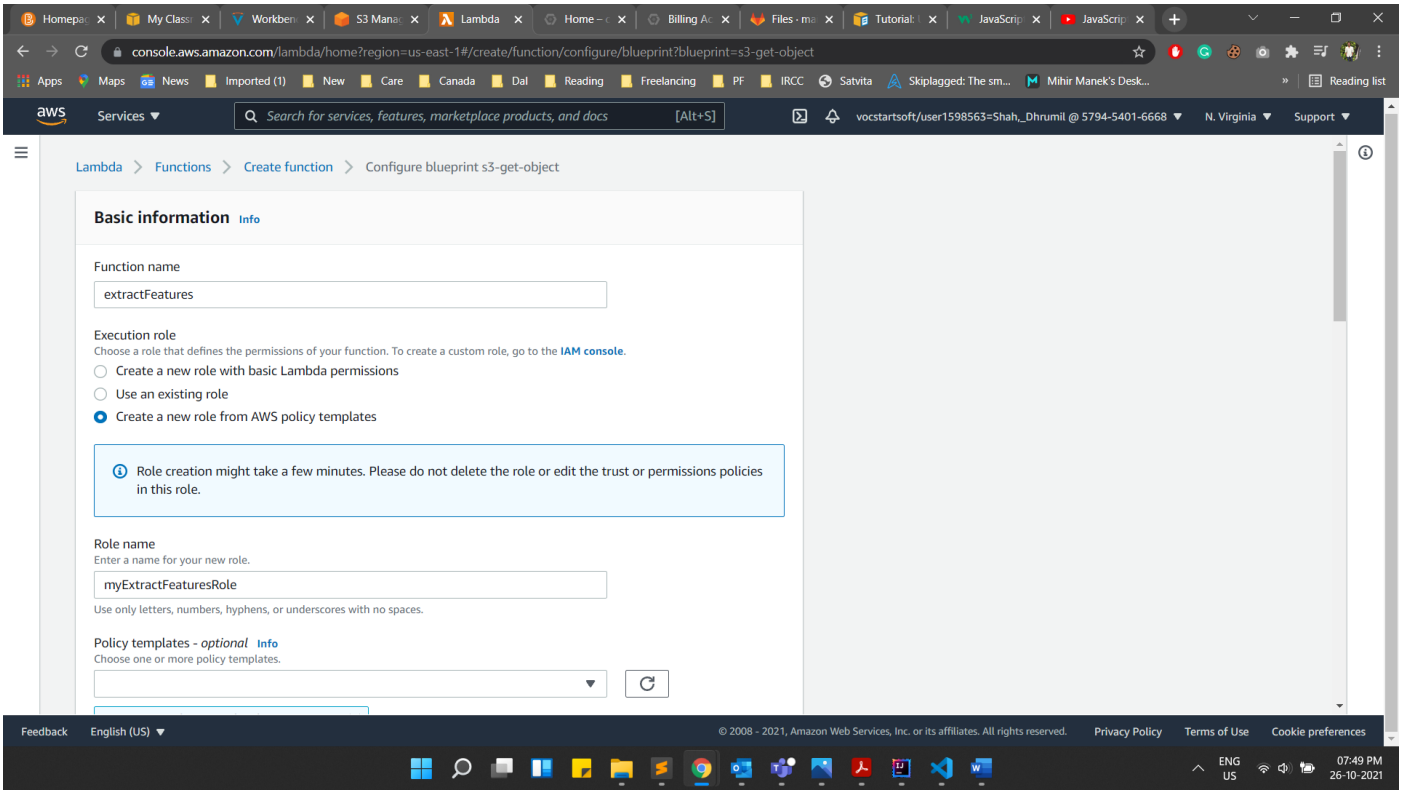

Figure 11: First Lambda Function (extractFeatures) 1

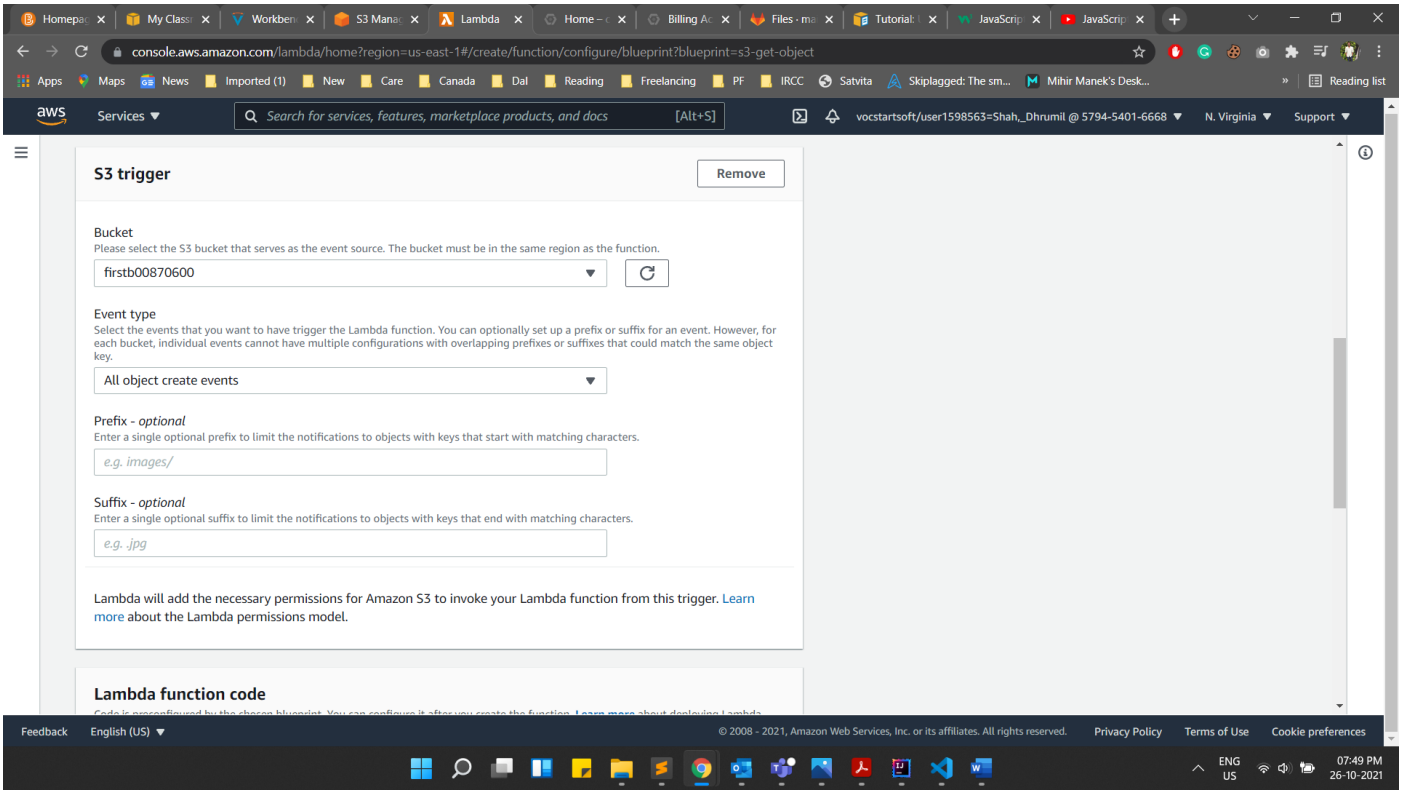

Figure 12: First Lambda Function (extractFeatures) 2

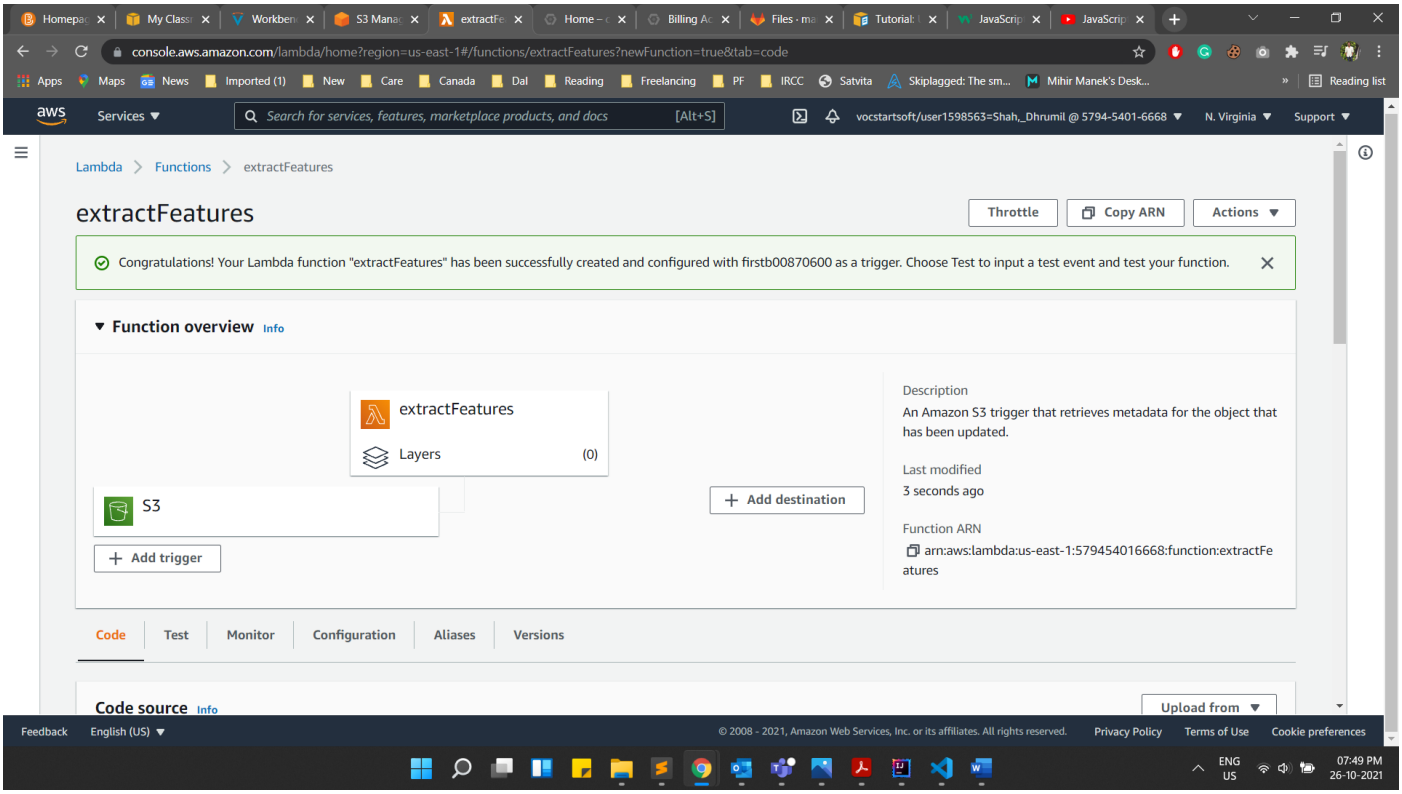

Figure 13: First Lambda Function (extractFeatures) 3

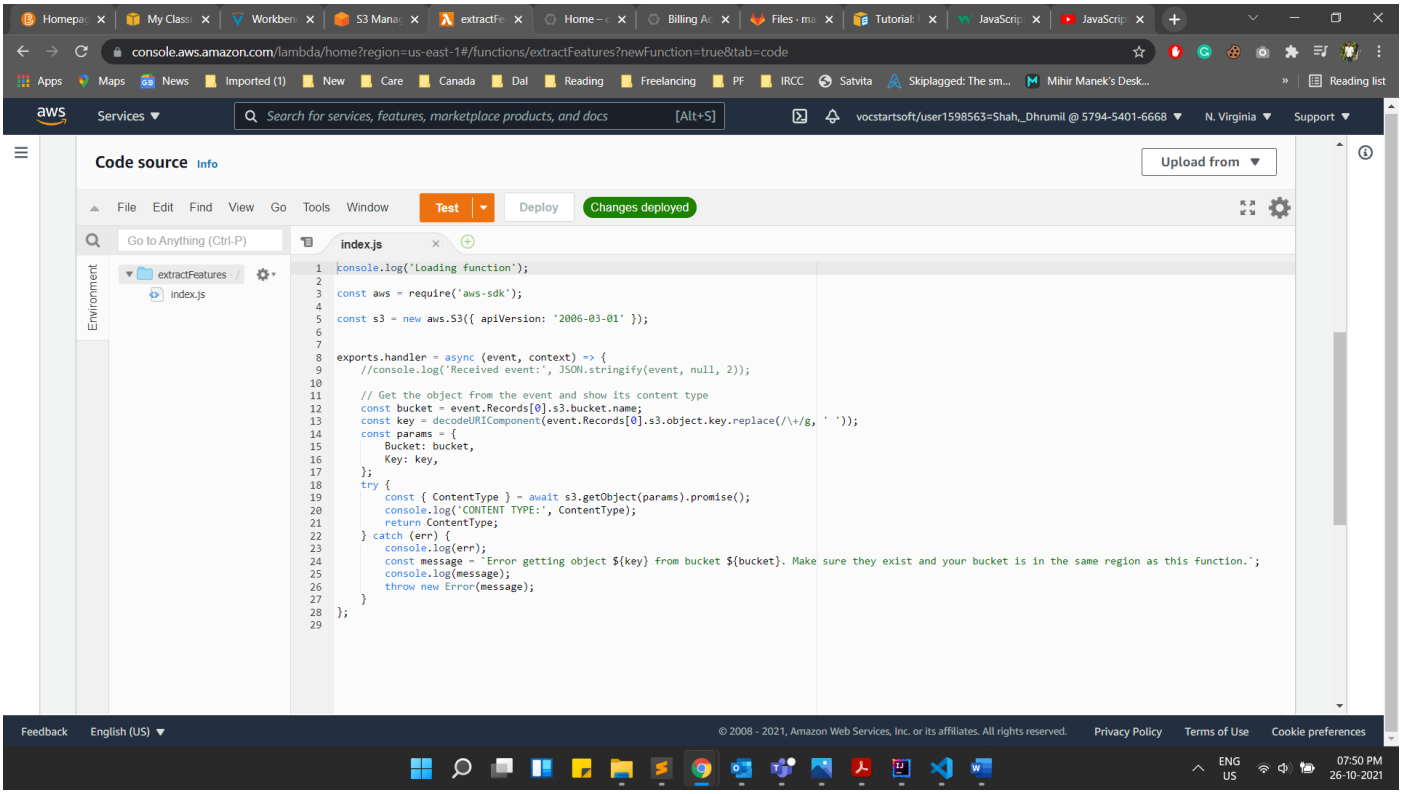

Figure 14: First Lambda Function (extractFeatures) 4

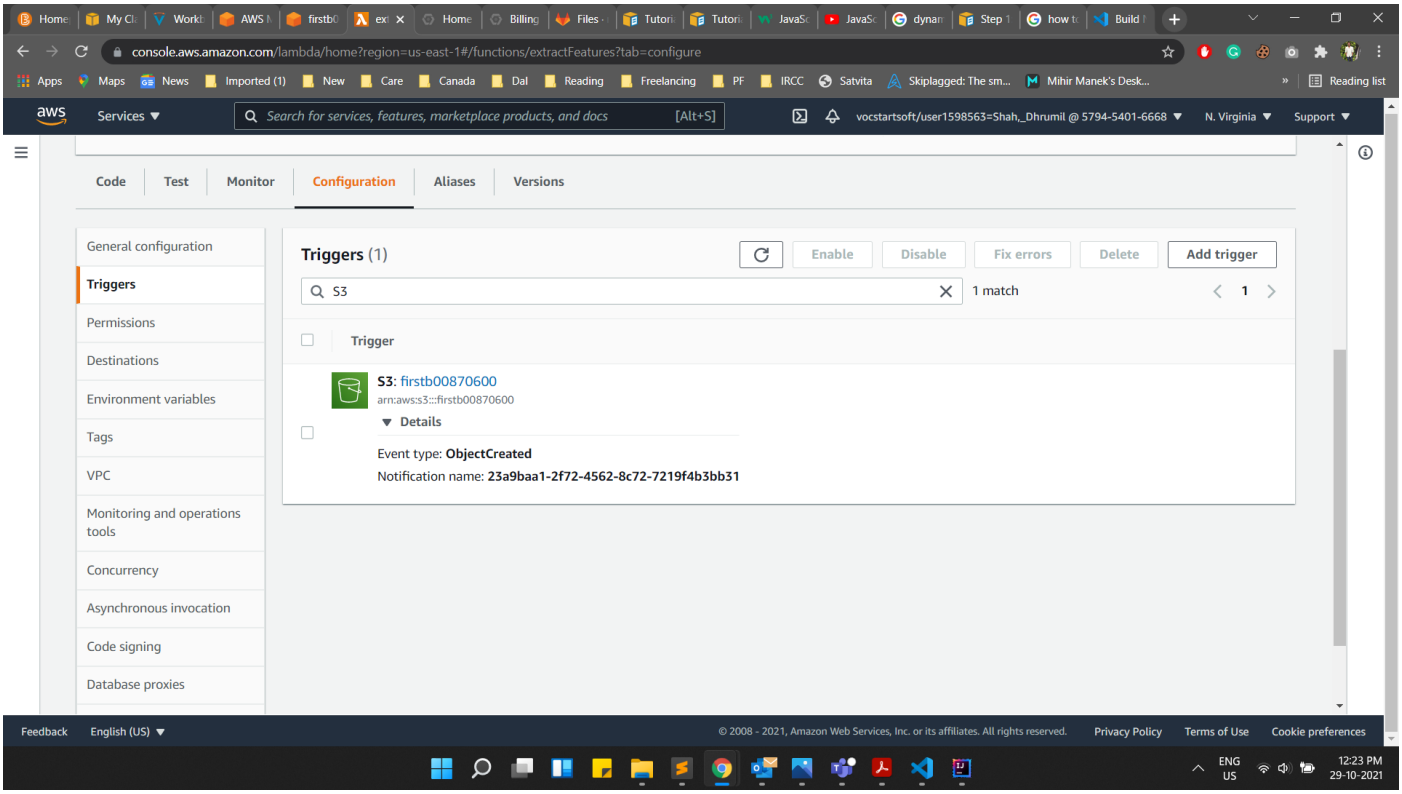

Figure 15: S3 Trigger added to the extractFeatures Lambda Function

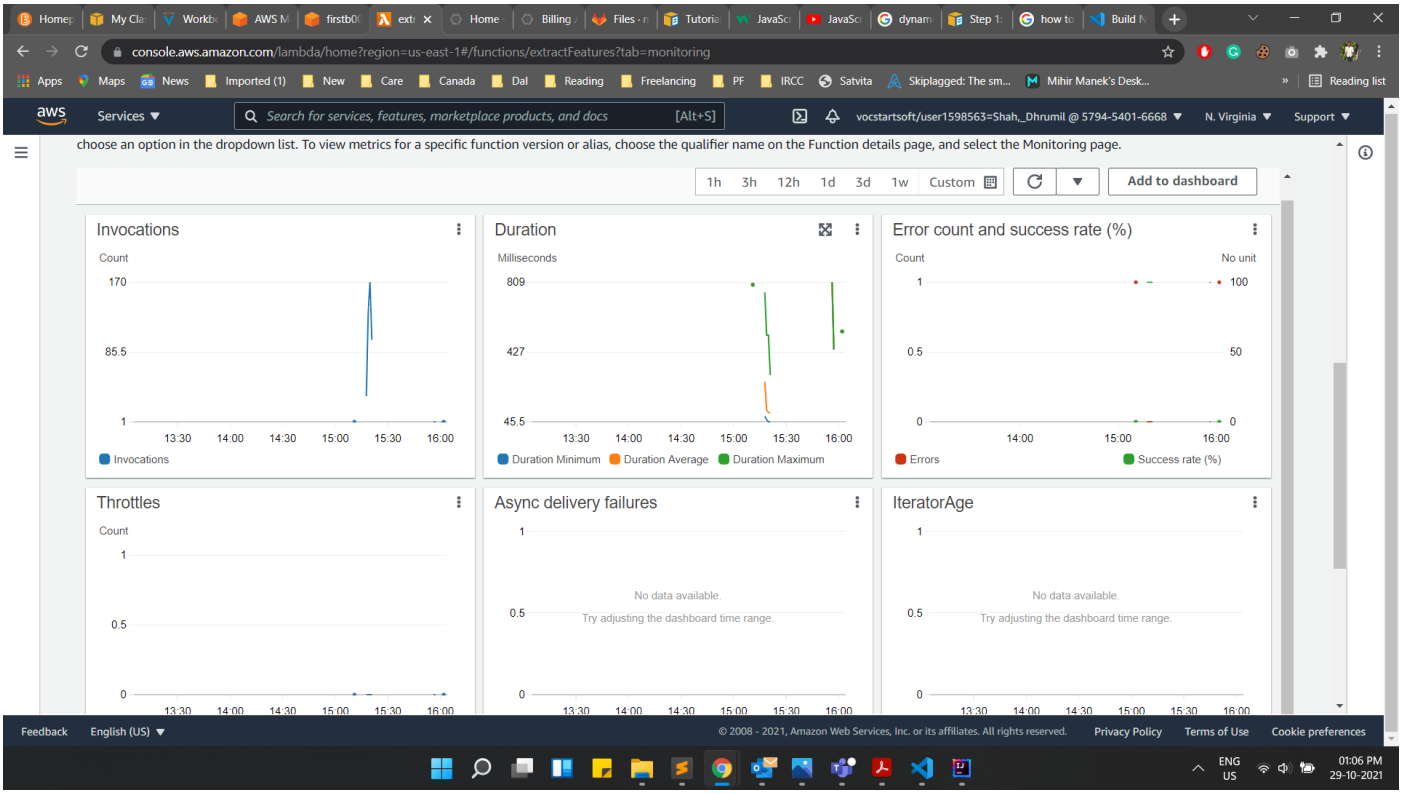

Figure 16: First Lambda Function (extractFeatures) – CloudWatch Logs Insights 1

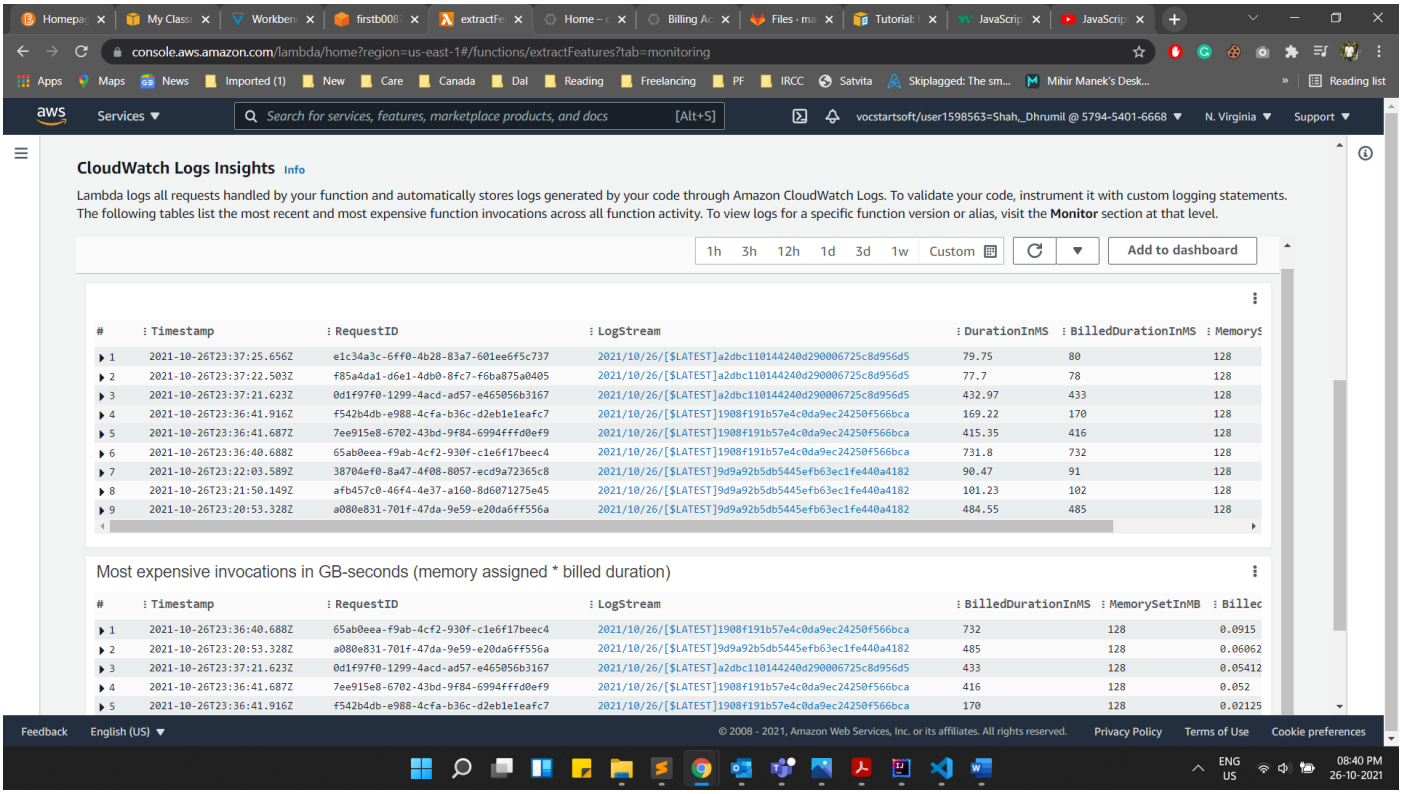

Figure 17: First Lambda Function (extractFeatures) – CloudWatch Logs Insights 2

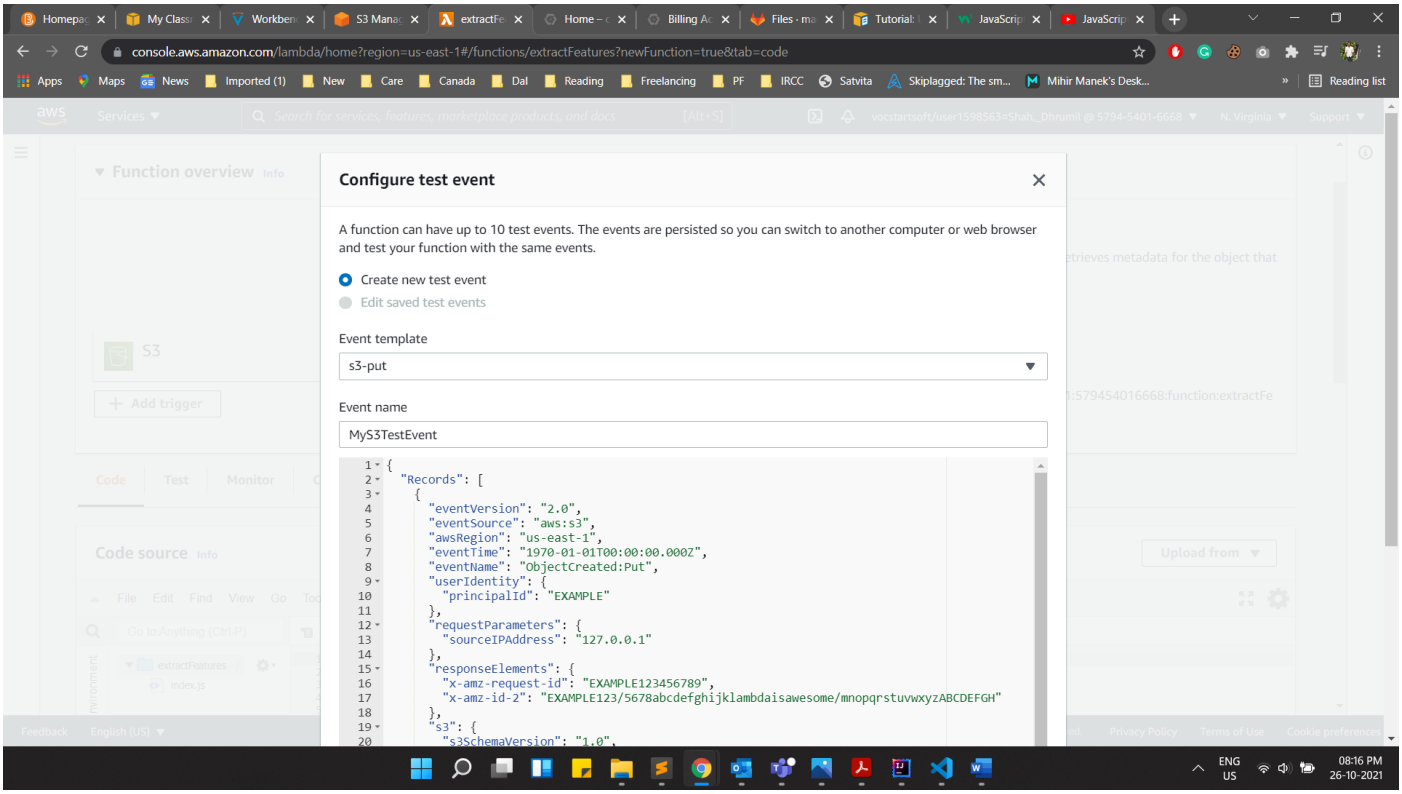

Figure 18: extractFeatures Lambda Function Configure Test Event 1

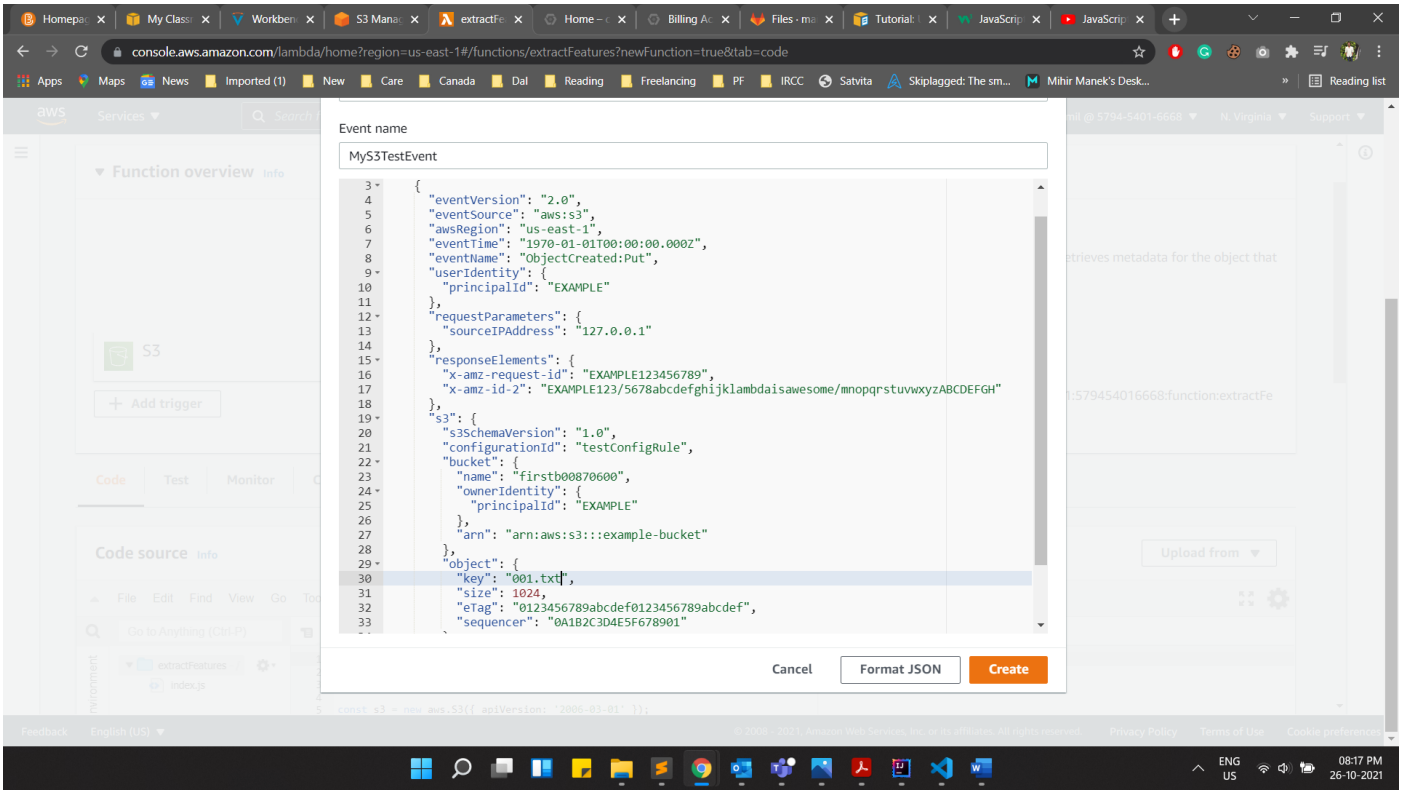

Figure 19: extractFeatures Lambda Function Configure Test Event 2

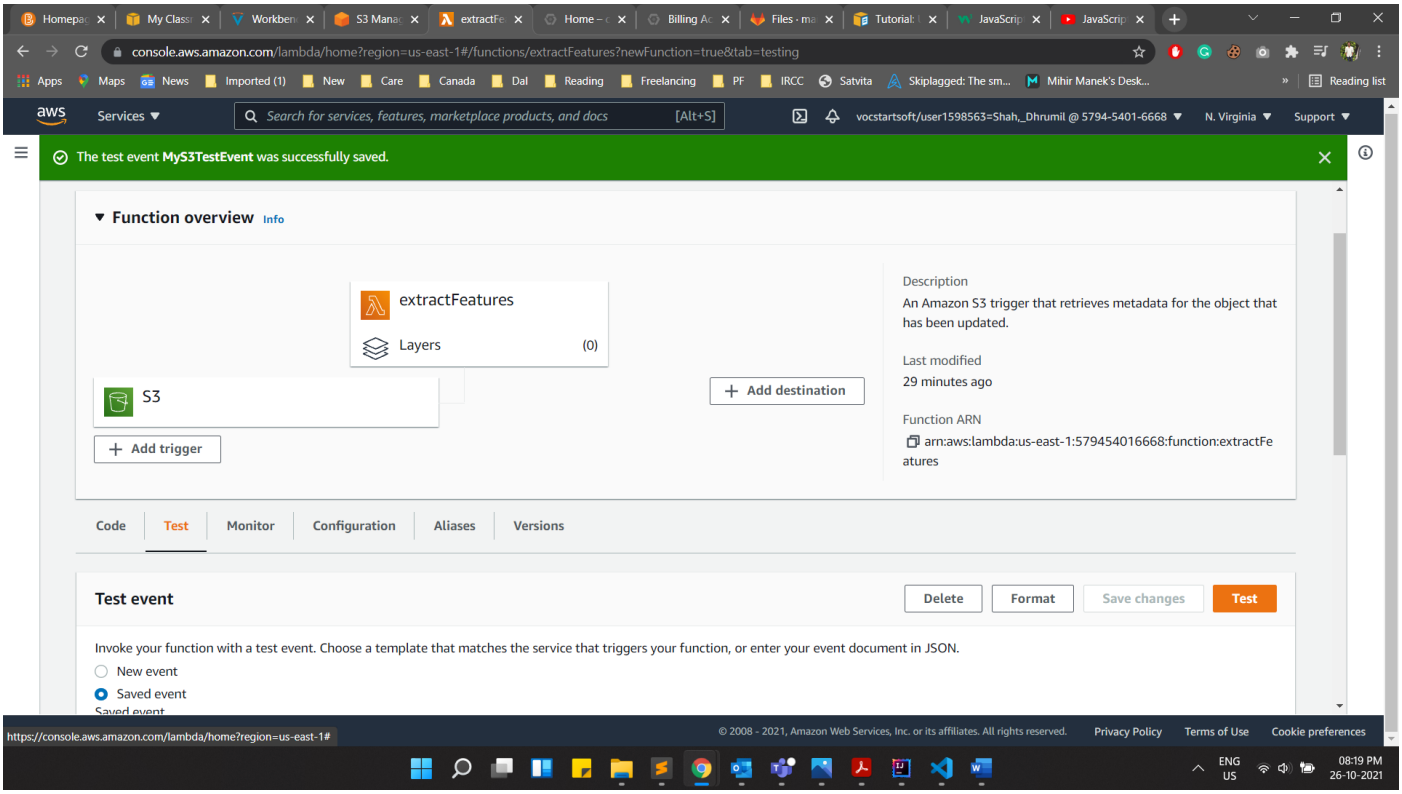

Figure 20: extractFeatures Lambda Function Configure Test Event 3

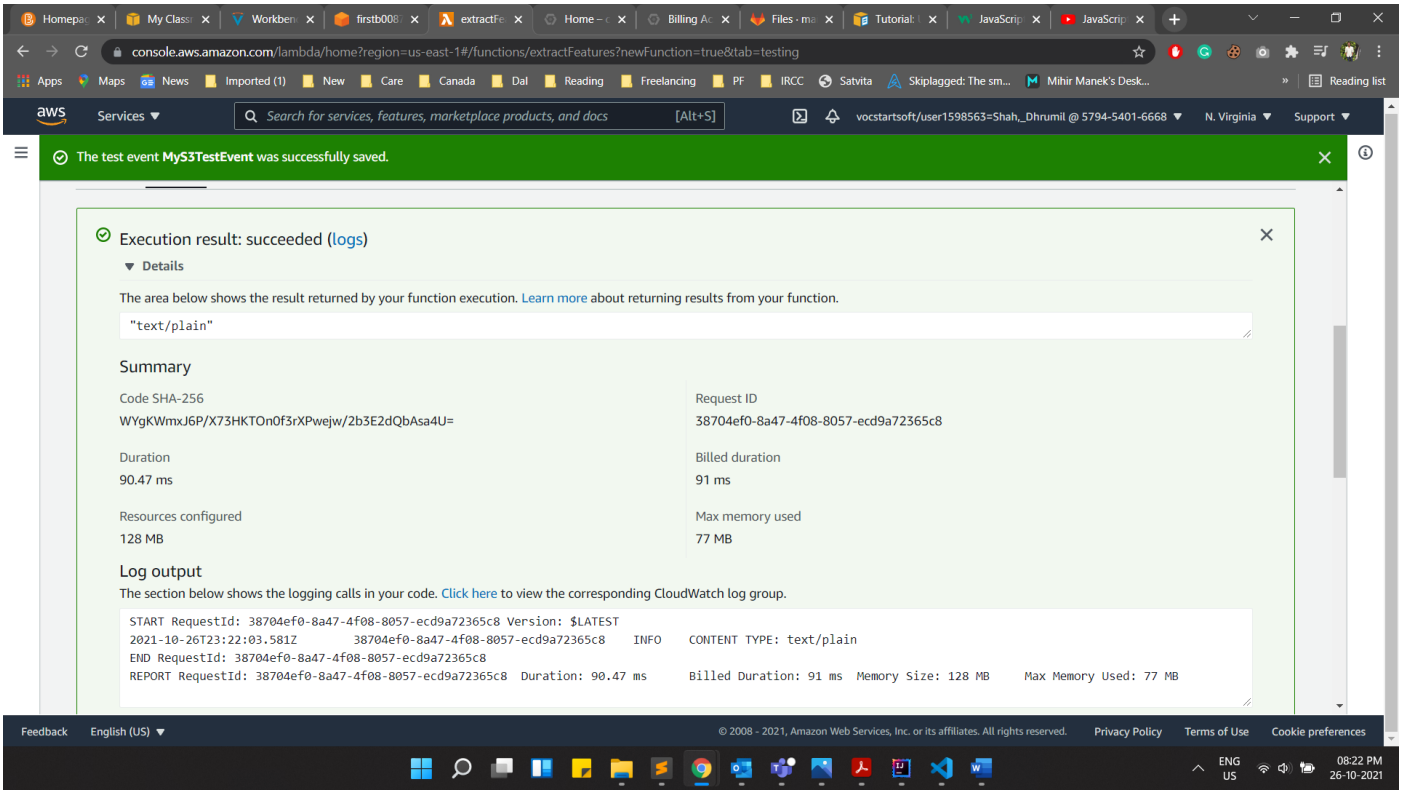

Figure 21: extractFeatures Lambda Function Configure Test Event 4

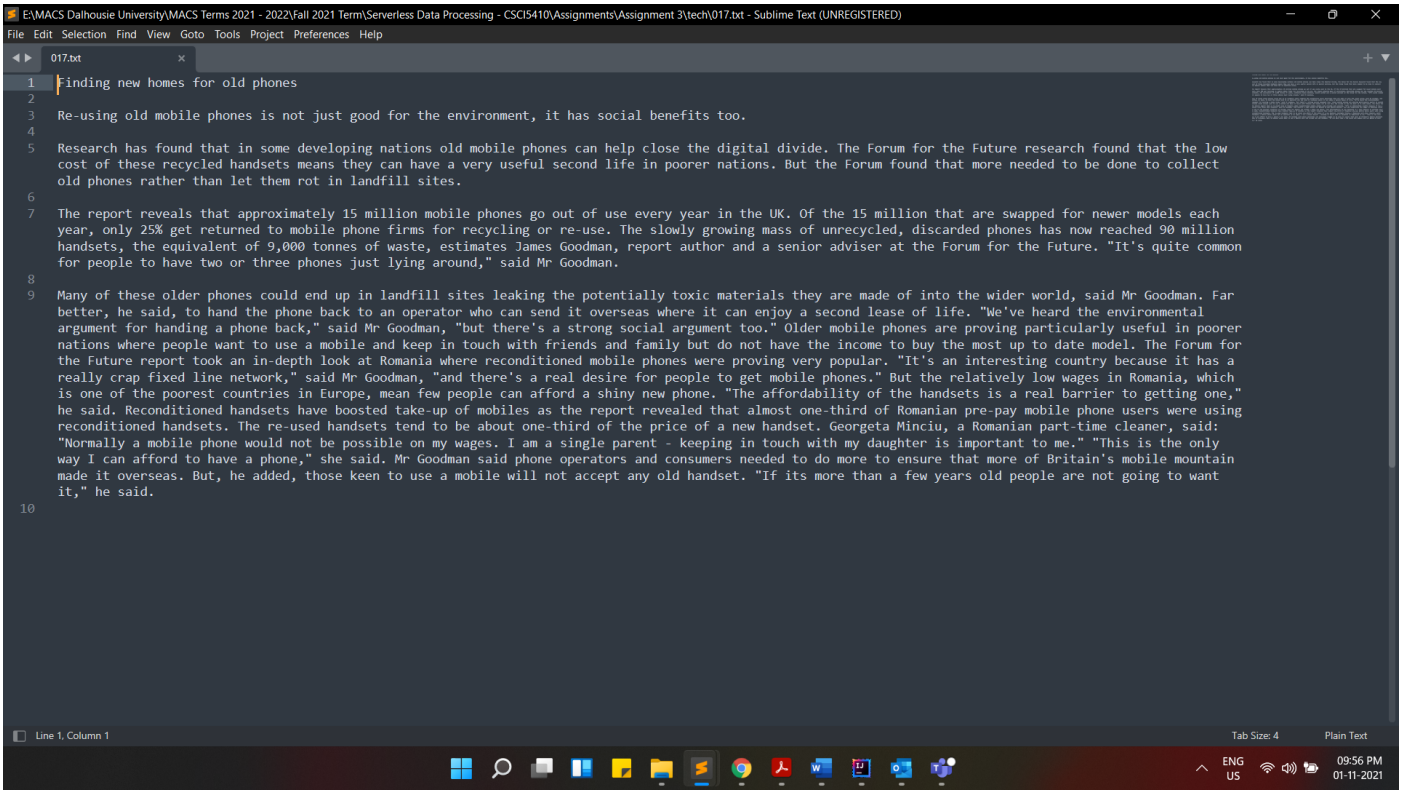

Figure 22: Original File Content

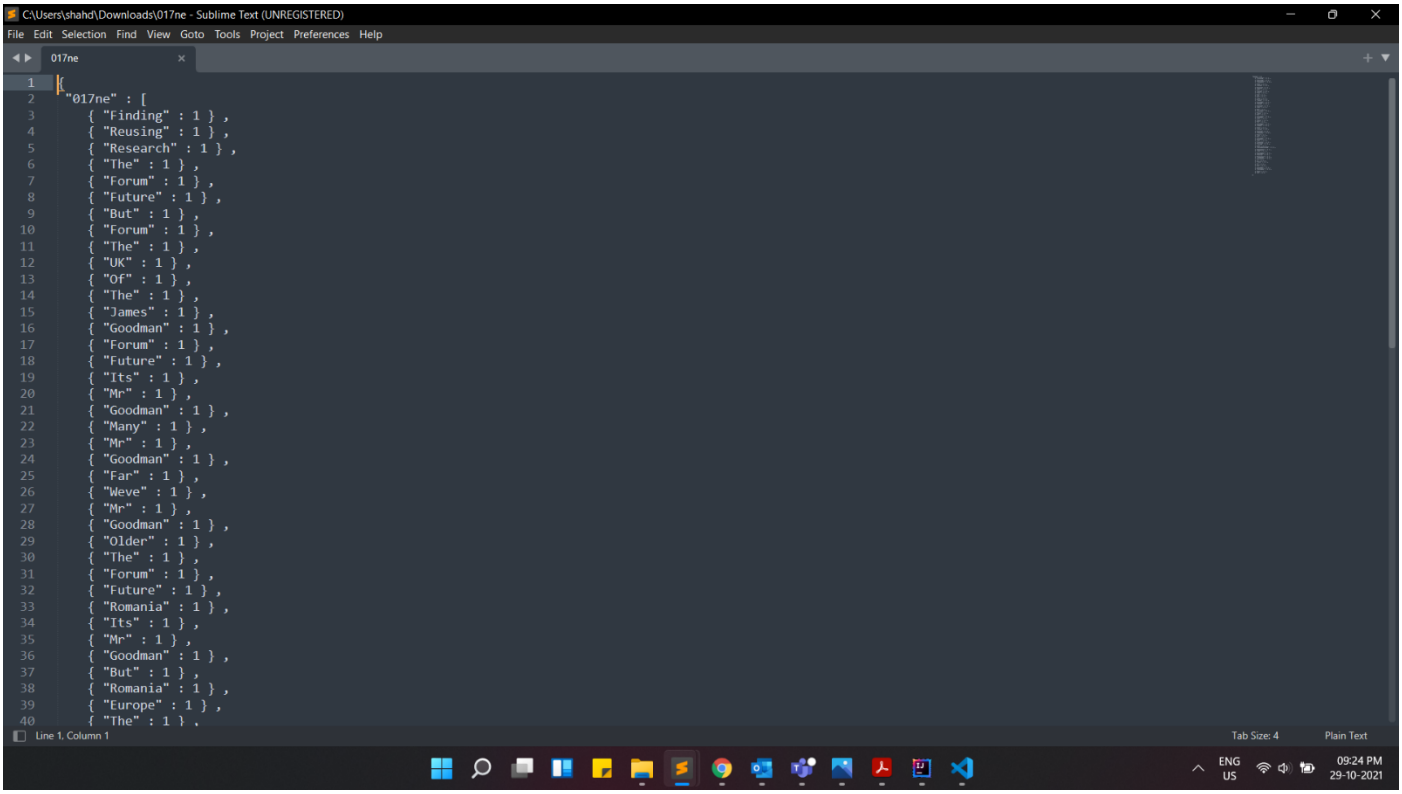

Figure 23: JSON File Created by Extracting Named Entites by extractFeatures Lambda Function

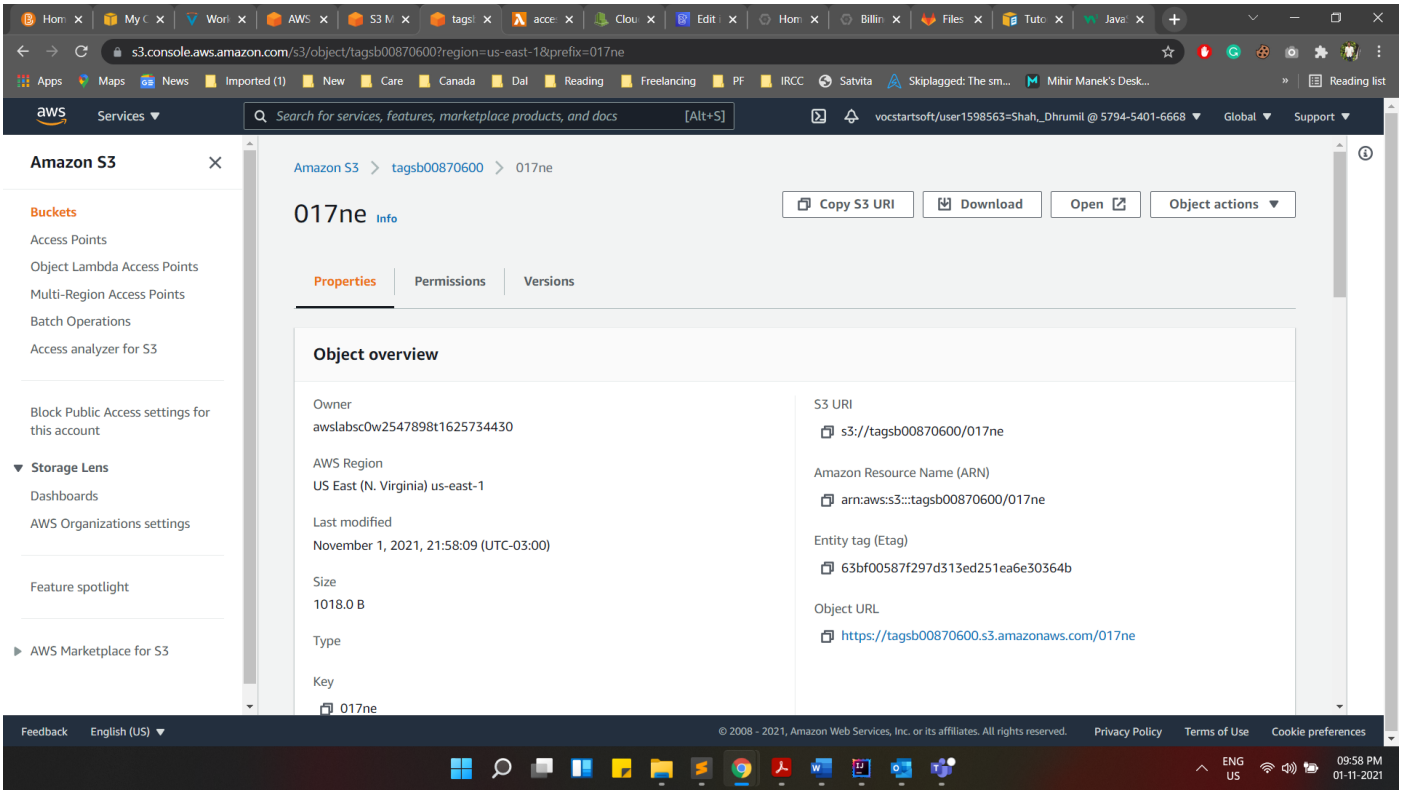

Figure 24: JSON File in tagsb00870600 S3 Bucket

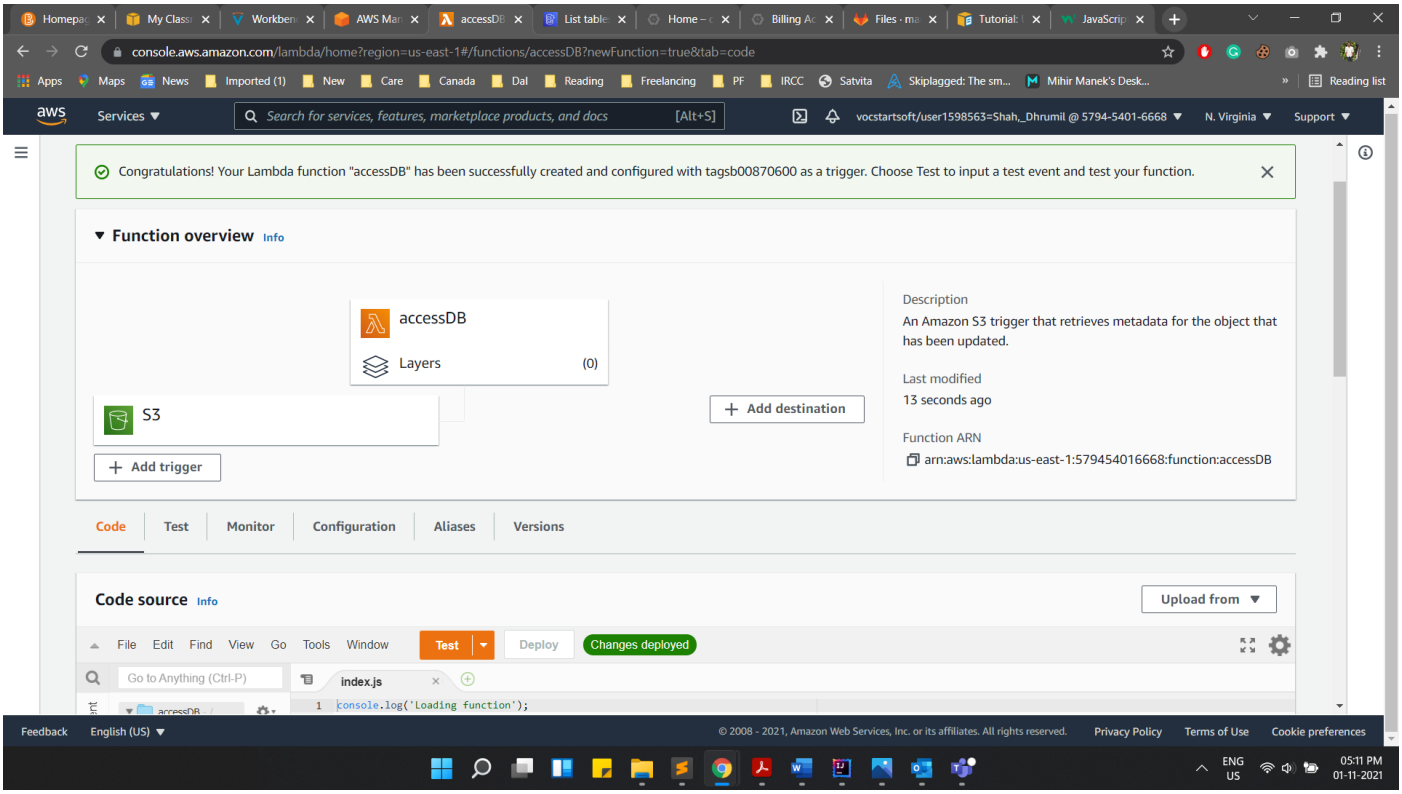

Figure 25: accessDB Lambda Function

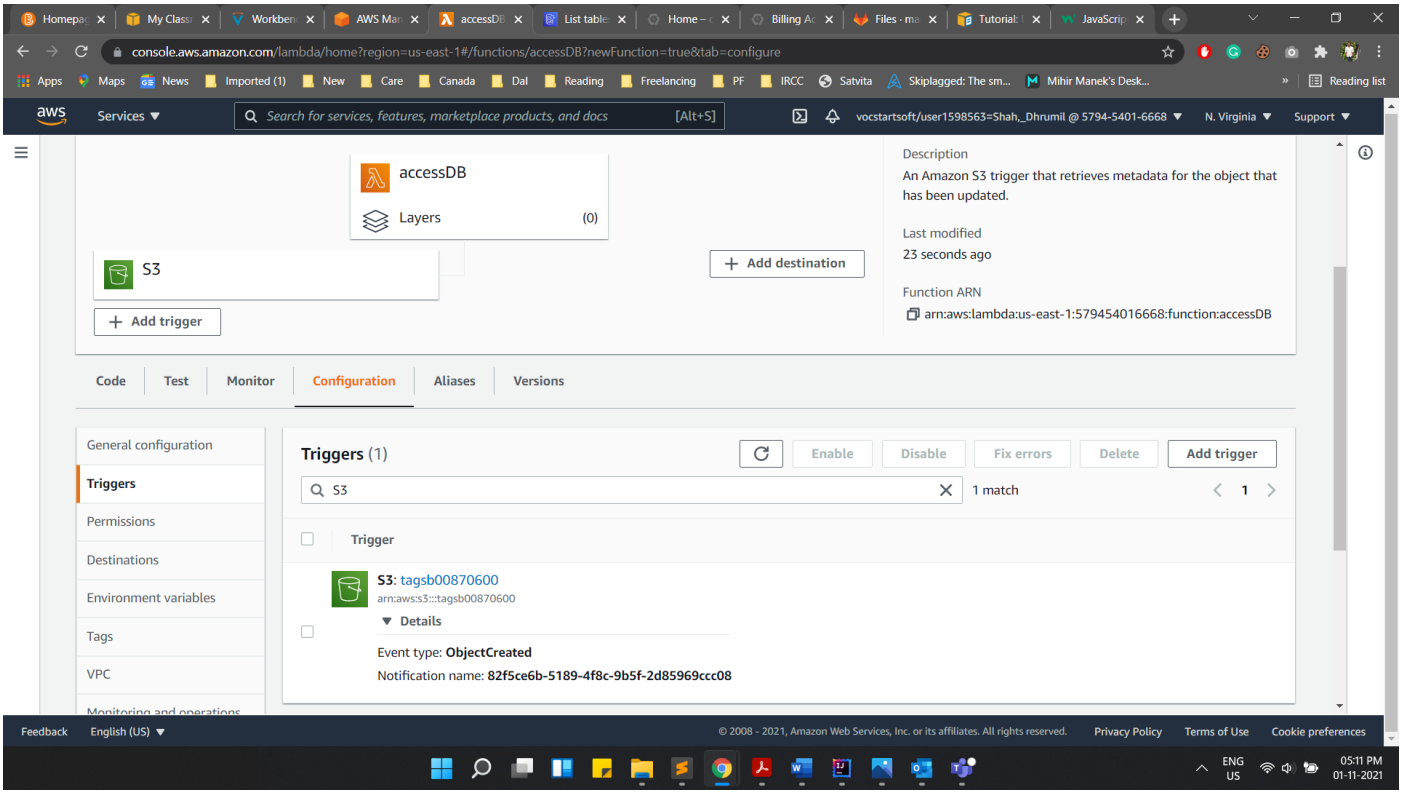

Figure 26: accessDB Lambda Function with S3 Bucket Trigger (tagsb00870600)

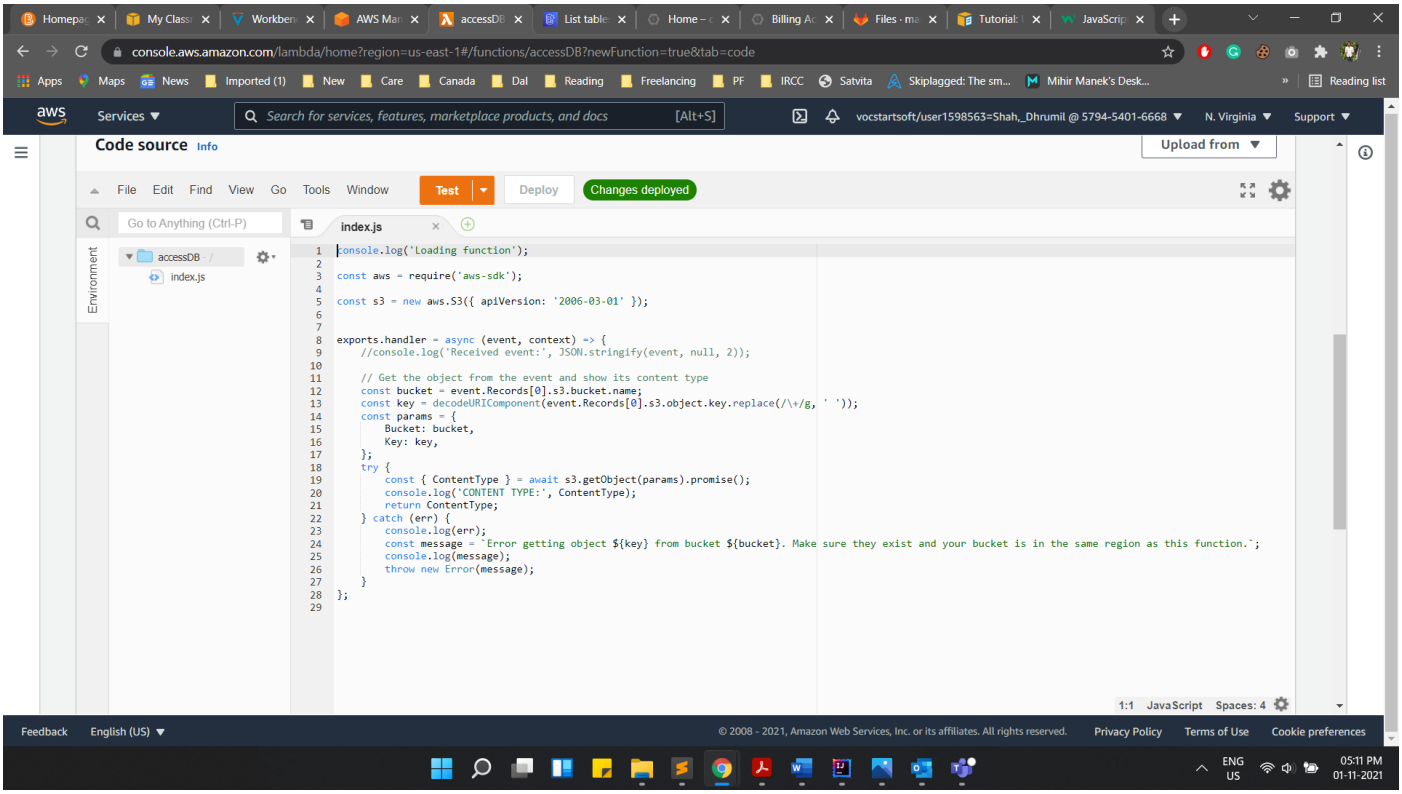

Figure 27:accessDB Lambda Function Boilerplate Code

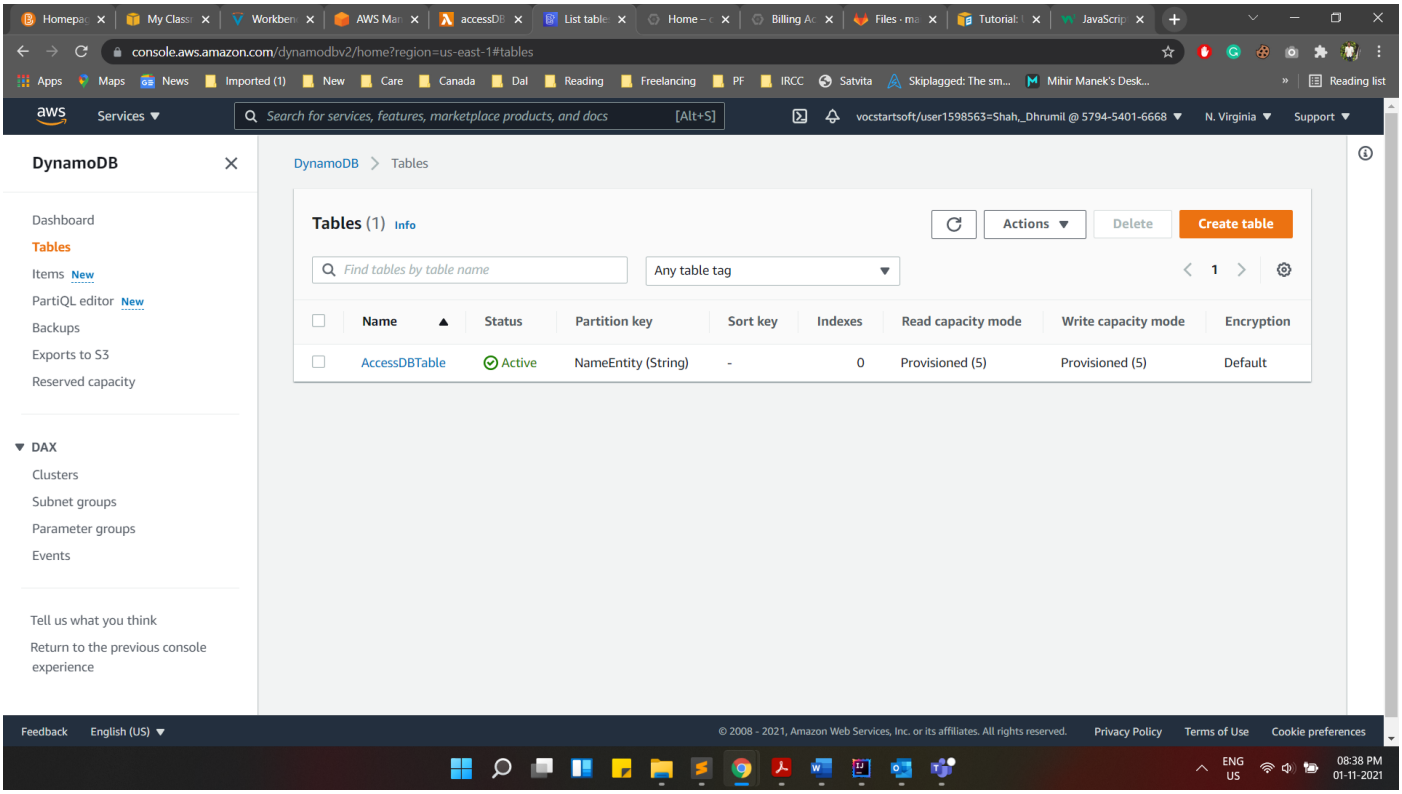

Figure 28: DynamoDB Table (AccessDBTable) 1

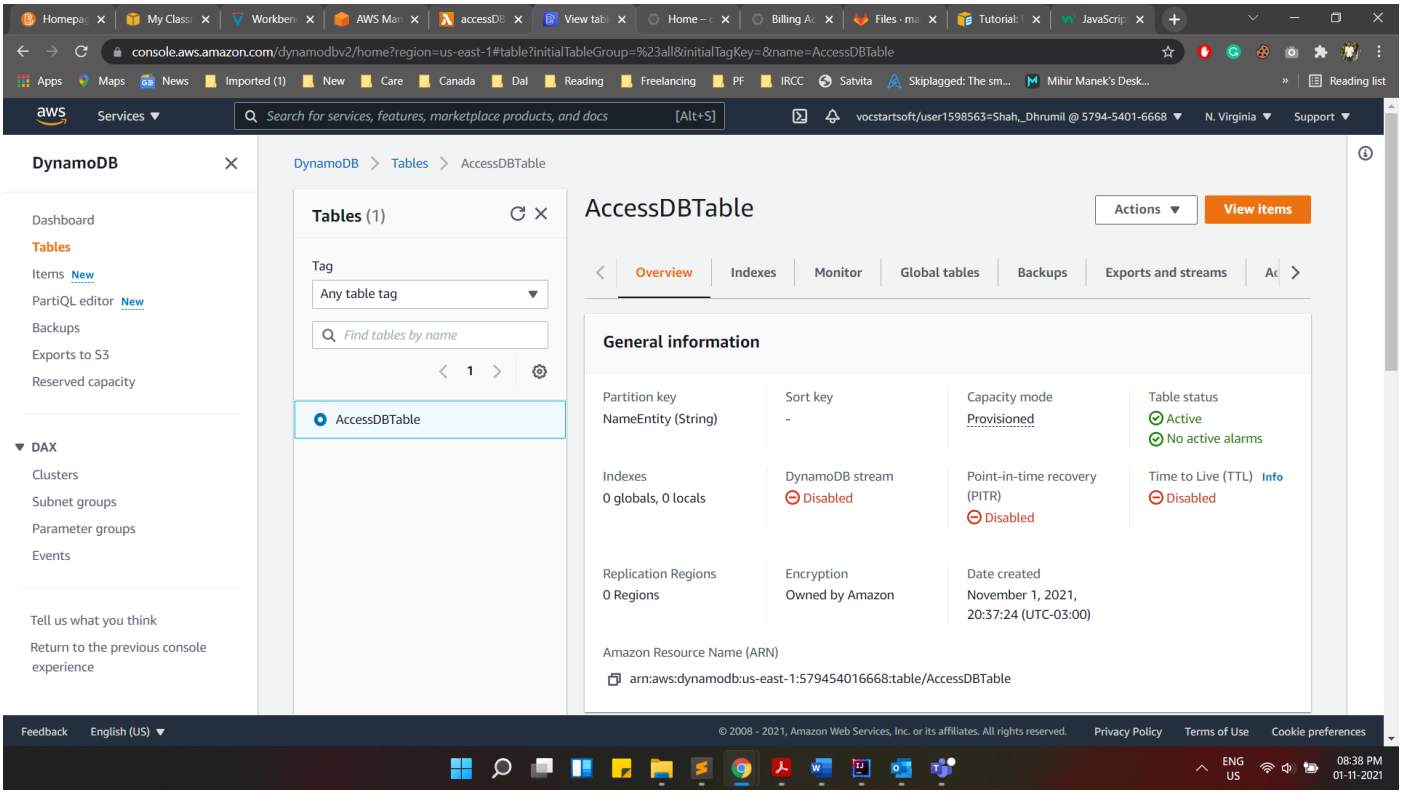

Figure 29: DynamoDB Table (AccessDBTable) 2

Code Snippets of the Part A:

Main.java:

```
import com.amazonaws.services.s3.AmazonS3;

import java.util.Scanner;

/**
 * Author: Dhrumil Rakesh Shah
 * Version: 1.0
 * Class: The Main class containing the boilerplate code of AWS SDK for Java
 * for Assignment 3
 */
public class Main {
  // The driver method
  public static void main(String[] args) {

    try {

      // Instantiating the Scanner object to read user input
      Scanner sc = new Scanner(System.in);

      // Declaring & Initializing the choice to 0 that stores the
      // choice made by the user
      int choice = 0;

      S3Task s3TaskObject = new S3Task();

      AmazonS3 s3Object = s3TaskObject.getConnection();
```

```
      // Looping through the menu
      while (choice != -1) {
        System.out.println("Choose any of the below tasks.");
        System.out.println("1. Create two new buckets.");
        System.out.println("2. Upload all files in tech folder to the " +
            "firstb00870600 bucket.");
        System.out.println("3. Testing the extractFeatures Lambda function.");
        System.out.println("4. Exit.");

        // Reading the user entered choice
        choice = sc.nextInt();

        // Switching through the different choices
        switch (choice) {
          // Creating a new bucket
          case 1:
            s3TaskObject.createBucket("firstb00870600", s3Object);
            s3TaskObject.createBucket("secondb0870600", s3Object);
            break;
          case 2:
            // Uploading the file to the created bucket
            s3TaskObject.fileUploadToS3Bucket(s3Object,
                "firstb00870600");
            break;
          case 3:
            // Testing the extractFeatures Lambda function
            s3TaskObject.testLambda(s3Object);
          case 4:
            // Exiting the application
            System.exit(0);
          default:
            // Default switch case
            System.out.println("Enter a valid option.");
            break;
        }
      }
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }
  }
}
```

S3Task.java

```
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicSessionCredentials;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.PutObjectRequest;

import java.io.File;

/**
 * Author: Dhrumil Rakesh Shah
 * Version: 1.0
 * Class: The AWS SDK for Java helper class
 */
public class S3Task {

  // The method to get connection object of the AWS account
  public AmazonS3 getConnection() {
```

```
    // Storing the AWS credentials to establish the connection
    BasicSessionCredentials sessionCredentials = new BasicSessionCredentials(
        "ASIAYN2RGTSOBPUQ5DBV",
        "4vxZQggC1hSF7QblDgErSuMJ0e0Ab6GfknBvjNUg",
        "FwoGZXIvYXdzEIn//////////wEaDD+7t6vcfZOA5QcwKCK9AdS25YlT" +
            "jDdeanTXWoWVa8zMqIdPeqwgbpsfSVptCU+2bi0DBGkcFv5czXVtsyg9Wv615wt" +
            "c4mOKjYGkh9Uh6bhn6qLYshVu4EtXloqrD+Ye7xO8r3wUXrT2SydHJtmv1oWBcCr" +
            "9XoxY64vs9jJByICeL930Ozybwsn7hzZYeQ1weumC5zE2SL8UukXJrM/n0MVI9Pz" +
            "pPa57zlkmeOYIDOkjF82PgUxrGOUeH1Nnh73Z8RjT5aW/sPeg9dN5MCjv8IGMBjI" +
            "tHprRFHAYt+KUKGnwPxf5JzDDl2MX2BYaNDPVFrM4roZzpXDAR5gOAGSRJpGm");

    // Establishing connection with the Amazon S3 service
    // and storing it in the s3Object
    AmazonS3 s3Object = AmazonS3ClientBuilder.standard().withCredentials(
            new AWSStaticCredentialsProvider(sessionCredentials))
        .withRegion(Regions.US_EAST_1)
        .build();

    // Returning the s3Object
    return s3Object;
  }

  // The public method to create a new S3 bucket
  public void createBucket(String bucketName, AmazonS3 s3Object) {
    try {
      // Creating a new S3 bucket using the s3Object
      s3Object.createBucket(bucketName);

      // Printing to the console
      System.out.format("A new bucket '%s' is created.\n", bucketName);

      // Catching the exception
    } catch (Exception e) {
      System.err.println(e.getMessage());
    }
  }

  // The public method to upload multiple files to a S3 bucket
  public void fileUploadToS3Bucket(AmazonS3 s3Object,
                                   String bucketName) {
    try {
      // Looping through the files in tech folder
      for (int i = 1; i <= 401; i++) {

        // Declaring the filePrefix and fileName
        String filePrefix = "";
        String fileName = "";

        // Conditioning through the files in tech folder
        if (i < 10) {
          filePrefix = "00" + i;
          fileName = "tech/00" + i + ".txt";
        } else if (i > 9 && i < 100) {
          filePrefix = "0" + i;
          fileName = "tech/0" + i + ".txt";
        } else {
          filePrefix = "" + i;
          fileName = "tech/" + i + ".txt";
        }

        // Instantiating the PutObjectRequest class
        PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName,
```

```
            filePrefix, new File(fileName));

        // Uploading the file to the S3 bucket
        s3Object.putObject(putObjectRequest);

        // Printing to the console
        System.out.format("The file '%s.txt' is pushed to the '%s' bucket \n",
            filePrefix, bucketName);

        // Adding a delay of 200 milliseconds after adding a new file to the bucket
        Thread.sleep(200);

        // Printing that 200 milliseconds sleep is added
        System.out.println("Added delay of 200 millis");
      }
      // Catching the exception
    } catch (Exception e) {
      System.err.println(e.getMessage());
    }
  }

  // Method to test the working of extractFeatures Lambda function
  public void testLambda(AmazonS3 s3Client) {
    String bucketName = "firstb00870600";
    String filePrefix = "001";
    String fileName = "tech/001.txt";
    try {
      PutObjectRequest request = new PutObjectRequest(bucketName, filePrefix, new
File(fileName));
      s3Client.putObject(request);

      // Printing to the console
      System.out.format("The file '%s.txt' is pushed to the '%s' bucket \n",
          filePrefix, bucketName);
    } catch (Exception e) {
      System.err.println(e.getMessage());
    }
  }
}
```

extractFeatures.js

```
console.log('Loading function');

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

const s3 = new aws.S3({ apiVersion: '2006-03-01' });


exports.handler = async (event, context) => {
   //console.log('Received event:', JSON.stringify(event, null, 2));

   // Get the object from the event and show its content type
   const bucket = event.Records[0].s3.bucket.name;
   const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, ' '));
   const params = {
      Bucket: bucket,
```

```
        Key: key,
    };
    console.log(params);
    extractFeatures(params);
    try {
        const { ContentType } = await s3.getObject(params).promise();
        console.log('CONTENT TYPE:', ContentType);
        return ContentType;
    } catch (err) {
        console.log(err);
        const message = `Error getting object ${key} from bucket ${bucket}. Make sure they exist and your
bucket is in the same region as this function.`;
        console.log(message);
        throw new Error(message);
    }
};

const extractFeatures = (params) => {

    let uploadParams = {
        Bucket: 'tagsb00870600',
        Key: ''
    };

    let wordCount = '{ \n "' + params.Key + 'ne" : [ ';

    s3.getObject(params, function (err, metadata) {
        if (err && err.code === 'Not found') {
            console.log("Object not found");
        } else {
            let filedata = metadata.Body.toString('utf-8');
            filedata = filedata.split("\n");
            for (let i = 0; i < filedata.length; i++) {

                let eachwords = filedata[i].split(' ');
                for (let k = 0; k < eachwords.length; k++) {
                    if (eachwords[k] === '') {
                        continue;
                    }
                    eachwords[k] = eachwords[k].replace(/[^a-zA-Z ]/g, "");
                    if (eachwords[k].charAt(0) === eachwords[k].charAt(0).toUpperCase()) {
                        if (eachwords[k] === '') {
                            continue;
                        }
                        let wordformat = "\n\t{ ";
                        wordformat = wordformat + "" + eachwords[k] + "" : 1 } ,';
                        wordCount = wordCount + wordformat;
                    }
                }
            }
            wordCount = wordCount.slice(0, -1);
```

```
        wordCount = wordCount + " \n]}";
        let filename = params.Key + "ne";
        uploadParams.Key = filename;
        uploadParams.Body = wordCount;
        s3.upload(uploadParams, function (s3Err, data) {
            if (s3Err) {
                console.log(s3Err);
            }
            console.log(`File uploaded successfully at ${data.Location}`);
        });
        console.log(wordCount);
        let countjsonarray = JSON.parse(wordCount);
    }
  });
};
```

accessDB.js

```
console.log('Loading function');

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

const s3 = new aws.S3({ apiVersion: '2006-03-01' });

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

const ddb = new aws.DynamoDB({ apiVersion: '2012-08-10 '});


exports.handler = async (event, context) => {
  //console.log('Received event:', JSON.stringify(event, null, 2));

  // Get the object from the event and show its content type
  const bucket = event.Records[0].s3.bucket.name;
  const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, ' '));
  const params = {
    Bucket: bucket,
    Key: key,
  };
  let filedata;
  s3.getObject(params, function (err, metadata) {
    if (err && err.code === 'Not found') {
      console.log("Object not found");
    } else {
      filedata = metadata.Body.toString('utf-8');
    }
  })
  accessDB(filedata, params.Key);
  try {
```

```
        const { ContentType } = await s3.getObject(params).promise();
        console.log('CONTENT TYPE:', ContentType);
        return ContentType;
    } catch (err) {
        console.log(err);
        const message = `Error getting object ${key} from bucket ${bucket}. Make sure they exist and your
bucket is in the same region as this function.`;
        console.log(message);
        throw new Error(message);
    }
};


const accessDB = async (wordsarray, key1) => {

    console.log(wordsarray[key1]);
    for (let a = 0; a < wordsarray[key1].length; a++) {
        for (var key in wordsarray[key1][a]) {
            docClient.put(nameEntityCol, function(err, data) {
    if (err) {
        console.error("Unable to add item. Error JSON:", JSON.stringify(err, null, 2));
    } else {
        console.log("Added item:", JSON.stringify(data, null, 2));
    }
    });
        docClient.put(frequencyCol, function(err, data) {
    if (err) {
        console.error("Unable to add item. Error JSON:", JSON.stringify(err, null, 2));
    } else {
        console.log("Added item:", JSON.stringify(data, null, 2));
    }
    });
        docClient.put(timestampOfEntryCol, new AttributeValue(new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss.SSSSSS")
            .format(new Date())), function(err, data) {
    if (err) {
        console.error("Unable to add item. Error JSON:", JSON.stringify(err, null, 2));
    } else {
        console.log("Added item:", JSON.stringify(data, null, 2));
    }
    });
    }
}
};
```

References:

1. https://docs.aws.amazon.com/lambda/latest/dg/with-s3-example.html
2. https://www.w3schools.com/js/js_json.asp
3. https://aws.amazon.com/education/awseducate/