CSCI 5410 – SERVERLESS DATA PROCESSING

Assignment – 5
Part - B

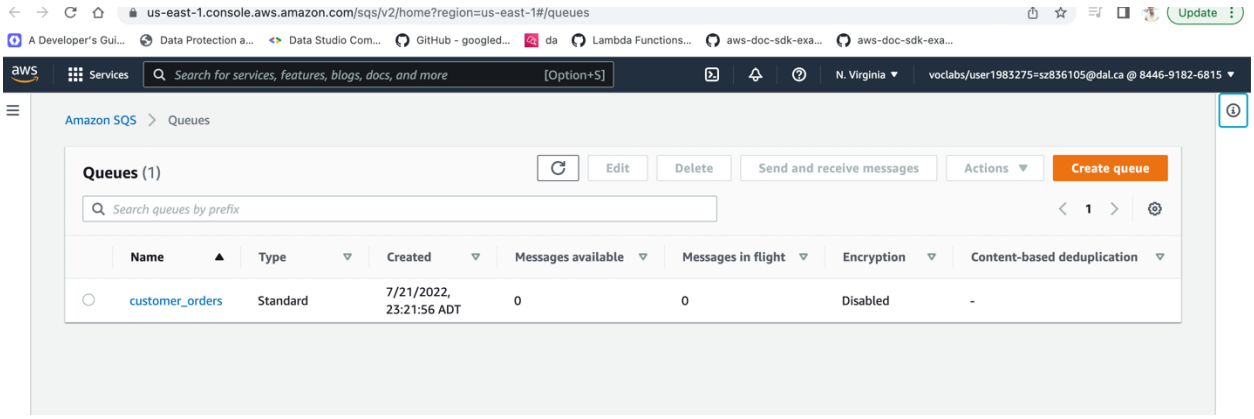
Name: Sai Vikas Chinthirla
Dal Id: B00911631
Instructor: Saurabh Dey

Summary on how the task is achieved

Using AWS SDK created a program to create a queue named “customer_orders”. Once the queue is created written a java code to send requests to queue with details like “Vehicle Type”, “Vehicle” and “Random Date”. To achieve this first defined a list of vehicles and vehicle types and by using random function randomly picked them. Next, once the request is sent to SQS a lambda is triggered and created the same with name as “poll_customer_orders”. In this lambda we will be accessing the details in the request from SQS and form the message details to post to SNS. For this create a SNS topic using AWS SDK programmatically and subscribed my dal email id to it. Finally, once the message is built it will be posted to SNS topic which will trigger the email. Below are the screenshots of the step-by-step process.

Screenshot of created SQS Queue

Created the SQS Queue named “customers” using AWS SDK programmatically.

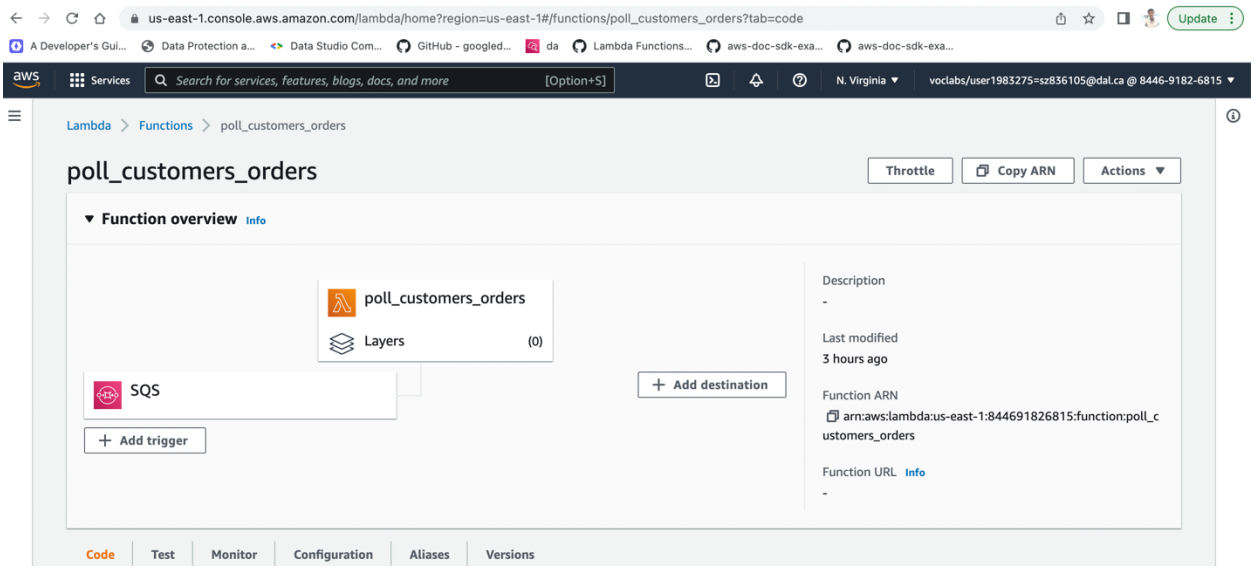


Program for creating SQS Queue:

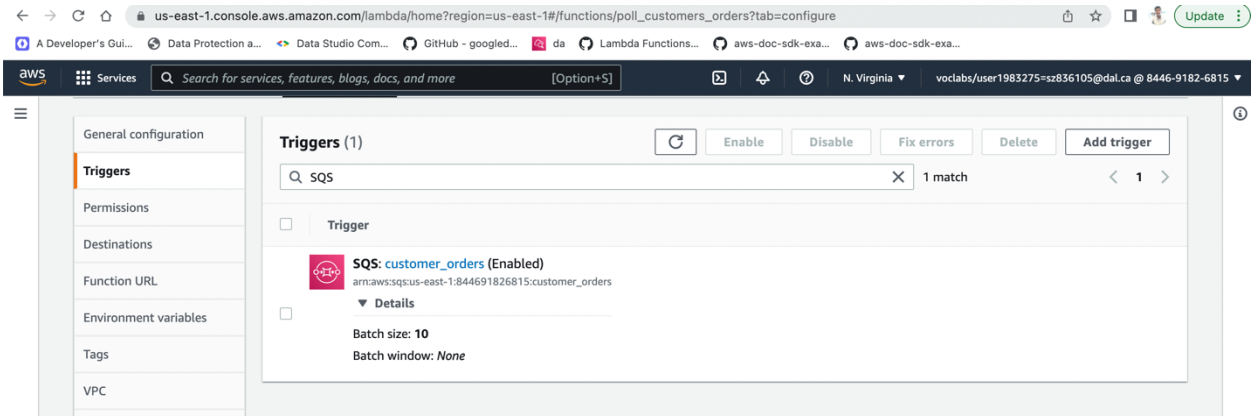
```
private static AmazonSQS createSQSQueue(AmazonSQS sqsQueue) {
    CreateQueueRequest createQueueRequest = new
CreateQueueRequest(QUEUE_NAME)
            .addAttributesEntry("DelaySeconds", "10")
            .addAttributesEntry("MessageRetentionPeriod", "86400");

    try {
        sqsQueue.createQueue(createQueueRequest);
    } catch (AmazonSQSException amazonSQSException) {
        throw amazonSQSException;
    }
    return sqsQueue;
}
```

Screenshot of creating Lambda Function

Created a lambda function named “poll_customer_orders” and configured SQS as trigger. As this lambda will be triggered every time a message a posted to SQS.



Added SQS Trigger:

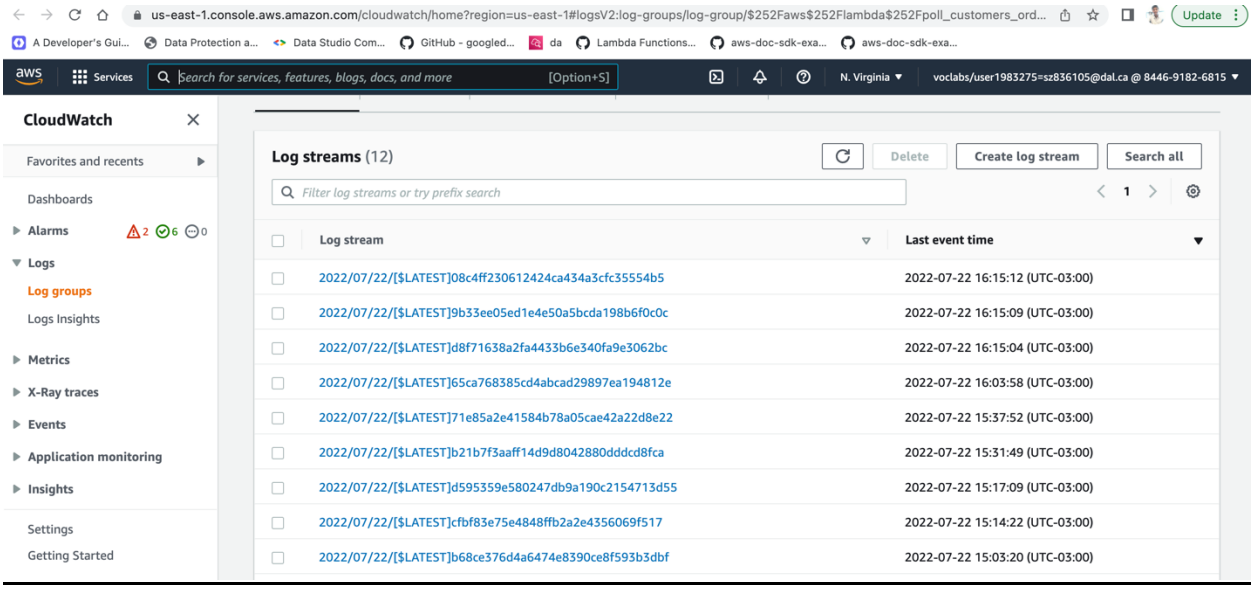


Screenshot of Script to send message to SQS

In the java program defined some random vehicles and vehicle types and by using random function picked one from each and sent message to SQS using message attributes.

```
private static void sendRequestToSQS(SqsClient sqs) {
    final Map<String, MessageAttributeValue> messageAttributes = new
HashMap<>();
    Random random = new Random();
    int index = random.nextInt(vehicleType.length);
    System.out.println(vehicleType[index]);

    messageAttributes.put("vehicleType", MessageAttributeValue.builder().
            dataType("String").stringValue(vehicleType[index]).build());
    messageAttributes.put("vehicle", MessageAttributeValue.builder()
            .dataType("String").stringValue(vehicle[index]).build());
    int day = generateRandomBetweenTwo(1, 28);
    int month = generateRandomBetweenTwo(1, 12);
    int year = generateRandomBetweenTwo(2022, 2023);
    String randomFutureDate = LocalDate.of(year, month, day).toString();
    messageAttributes.put("deliveryDate",
MessageAttributeValue.builder().dataType("String").stringValue(randomFutureDa
te).build());

    SendMessageRequest sendMsgRequest = SendMessageRequest.builder()
            .queueUrl("https://sqs.us-east-
1.amazonaws.com/844691826815/customer_orders")
            .messageBody("some request")
            .messageAttributes(messageAttributes)
            .delaySeconds(5)
            .build();
    sqs.sendMessage(sendMsgRequest);
}
```

Screenshot of Lambda Triggering By looking at Cloud Watch Logs



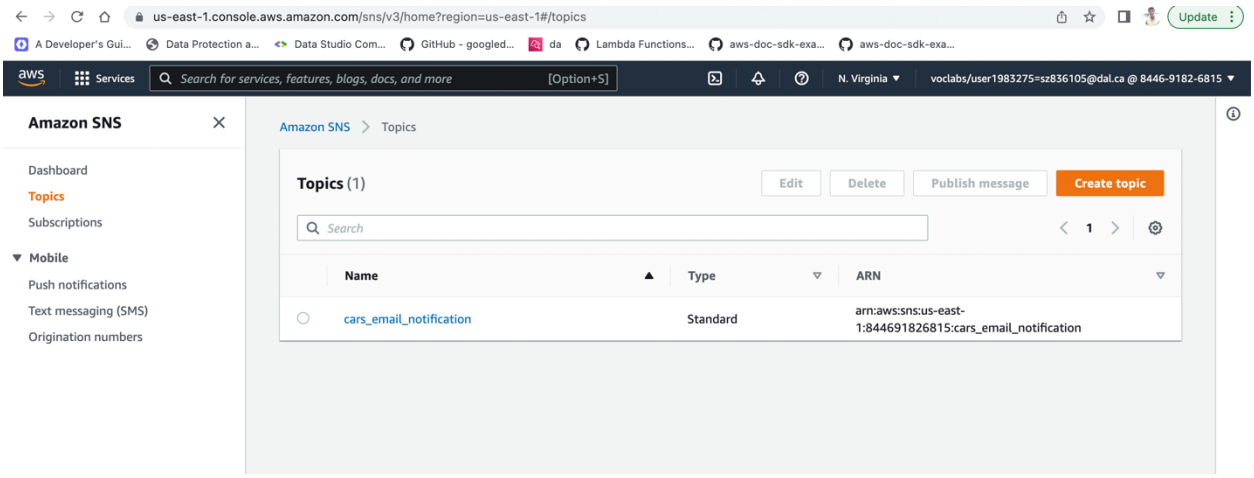
Once this lambda is triggered, we will pick all the order related details and form a message. This formed message will be posted to SNS topic. Below is the program which runs in lambda function when there is a message in SQS.

```
package lambda;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.MessageAttributeValue;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

import java.util.HashMap;
import java.util.Map;

public class CustomerOrdersPollingLambda implements RequestHandler<SQSEvent,
Void> {

    @Override
    public Void handleRequest(SQSEvent sqsEvent, Context context) {
        String vehicleType = "";
        String vehicle = "";
        String futureDate = "";
        for (SQSEvent.SQSMessage msg: sqsEvent.getRecords()) {
            vehicleType = new
String(String.valueOf(msg.getMessageAttributes().get("vehicleType").getString
```

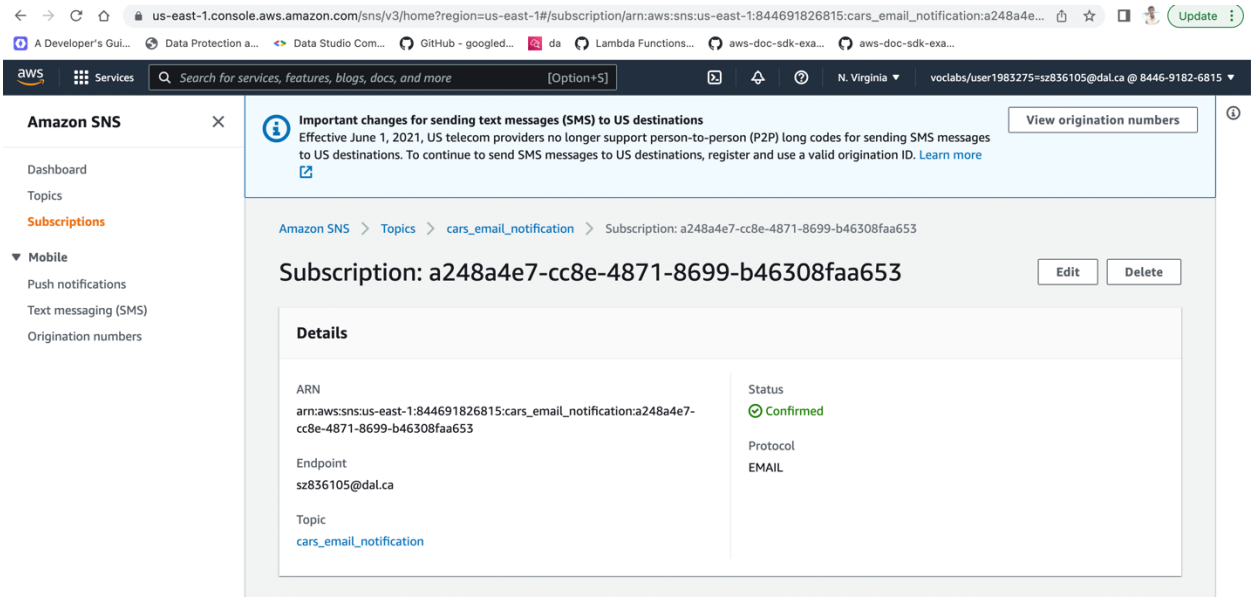
```
Value()));
            vehicle = new
String(String.valueOf(msg.getMessageAttributes().get("vehicle").getStringValu
e()));
            futureDate = new
String(String.valueOf(msg.getMessageAttributes().get("deliveryDate").getStrin
gValue()));
        }
        StringBuilder messageToDeliver = new StringBuilder();
        messageToDeliver.append("Vehicle Type :
").append(vehicleType).append("\n");
        messageToDeliver.append("Vehicle to deliver :
").append(vehicle).append("\n");
        messageToDeliver.append("Delivery Date :
").append(futureDate).append("\n").append("\n").append("Above are the details
for delivery.");
        SnsClient snsClient = SnsClient.builder().region(Region.US_EAST_1)
                .build();
       . Map<String, MessageAttributeValue> attributes = new HashMap<>();
        attributes.put("Vehicle_Type",
MessageAttributeValue.builder().dataType("String").stringValue("SUV").build()
);
        try {
//            PublishRequest request1 = PublishRequest.builder()
            PublishRequest request = PublishRequest.builder().subject("Car
Booking
Confirmation").message(messageToDeliver.toString()).messageAttributes(attribu
tes).
                    topicArn("arn:aws:sns:us-east-
1:844691826815:cars_email_notification").build();
            PublishResponse result = snsClient.publish(request);
            System.out.println(result.messageId() + " Message sent. Status is
" + result.sdkHttpResponse().statusCode());
        } catch (SnsException ex) {
            System.out.println(ex);
        }
        return null;
    }
}
```

Screenshot of created SNS Topic Programatically

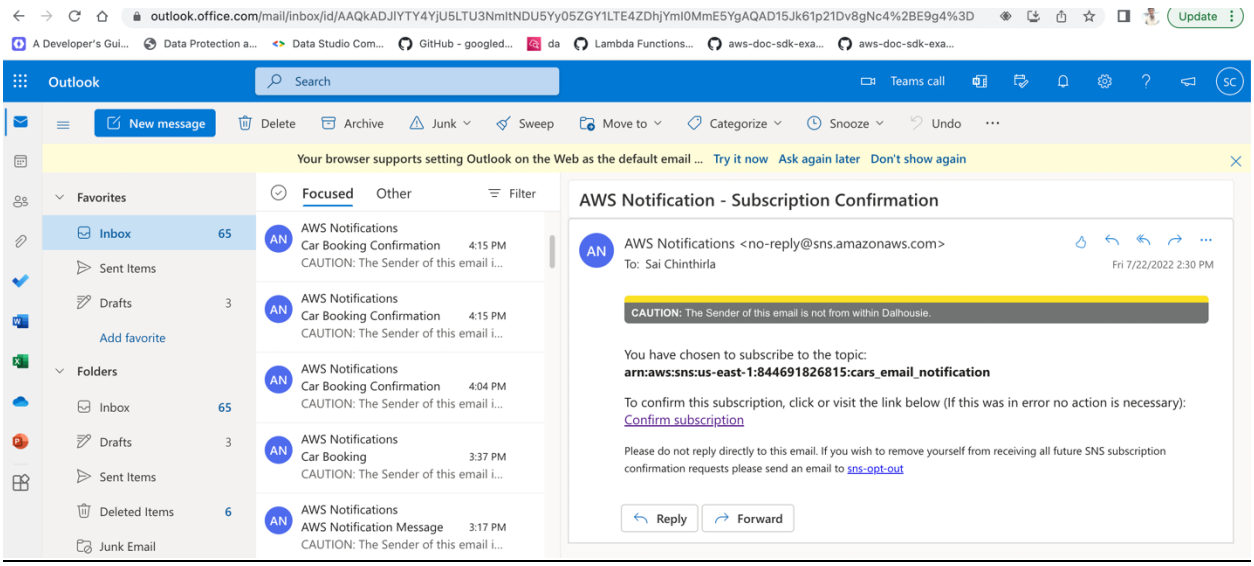
Using AWS SDK created a SNS topic “cars_email_notification” programmatically. And then added subscription with my dal email to it.



Screenshot of added Subscription to created topic



Screenshot of confirmation mail received



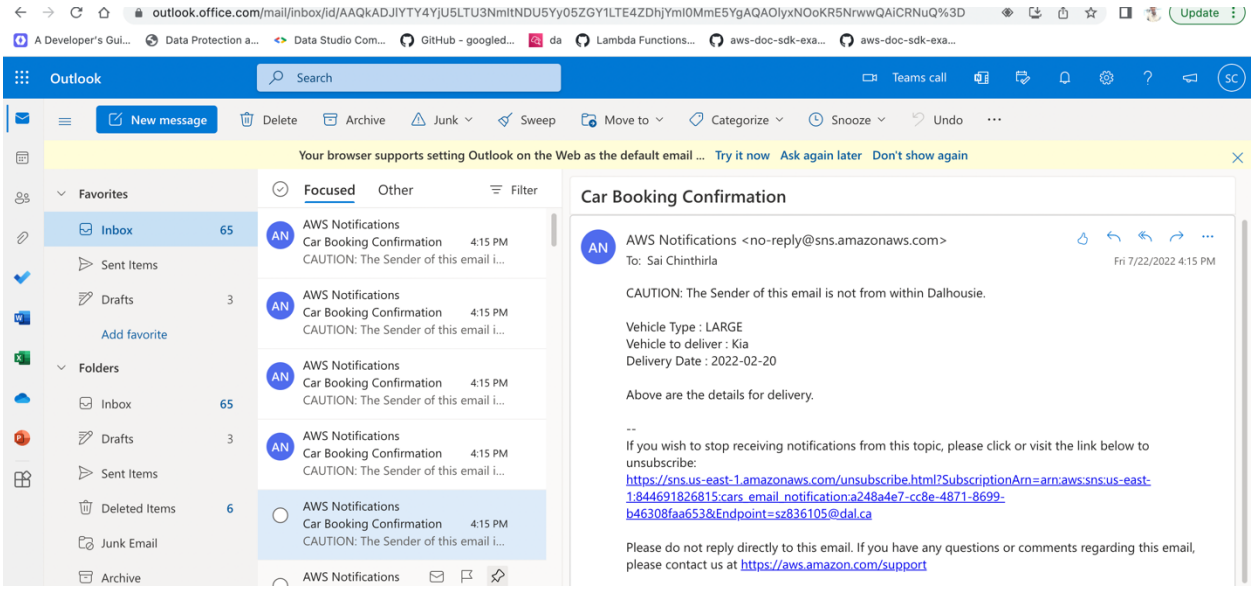
Program for creating SNS topic programmatically:

```
private static void createTopic(SnsClient snsClient) {
    CreateTopicResponse createTopicResponse = null;
    try {
        CreateTopicRequest createTopicRequest = CreateTopicRequest.builder()
                .name(TOPIC_NAME).build();
        createTopicResponse = snsClient.createTopic(createTopicRequest);
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

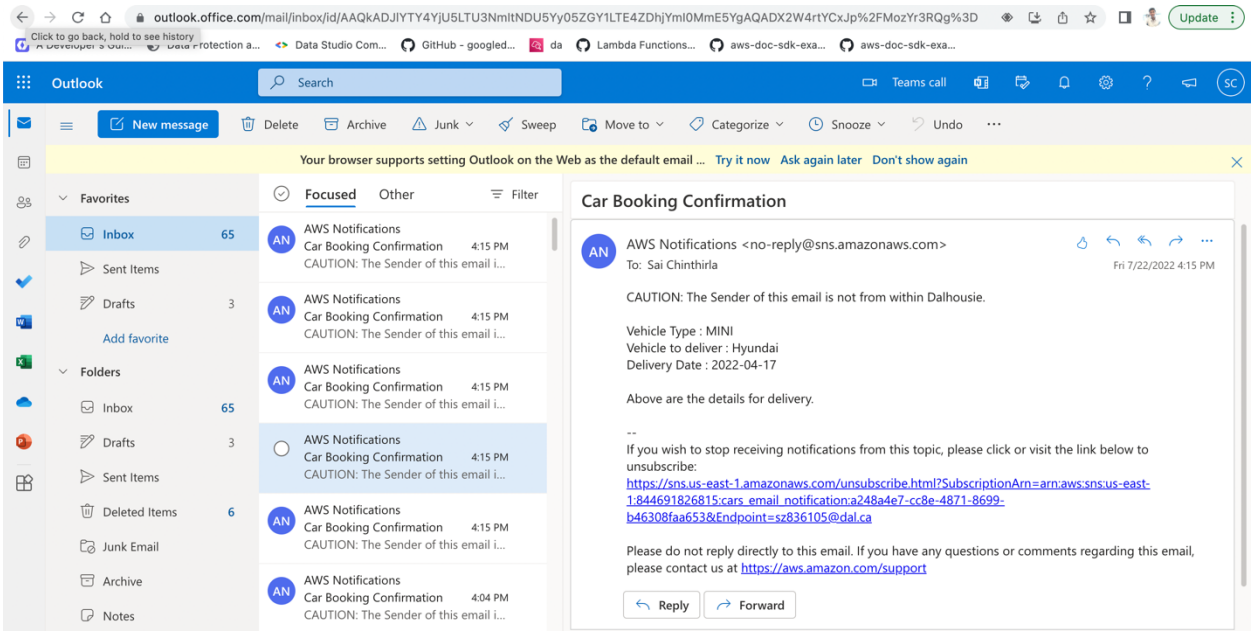
Once the SNS topic is created and subscribed. I have sent the request to SQS Queue. Which triggered lambda and lambda processed the request and published message to SNS topic. Once the message is published email is triggered. I have triggered 5 requests from the program to SQS and 5 emails related to those are triggered. Below are the screenshots of triggered emails.

Screenshot of Triggered Emails

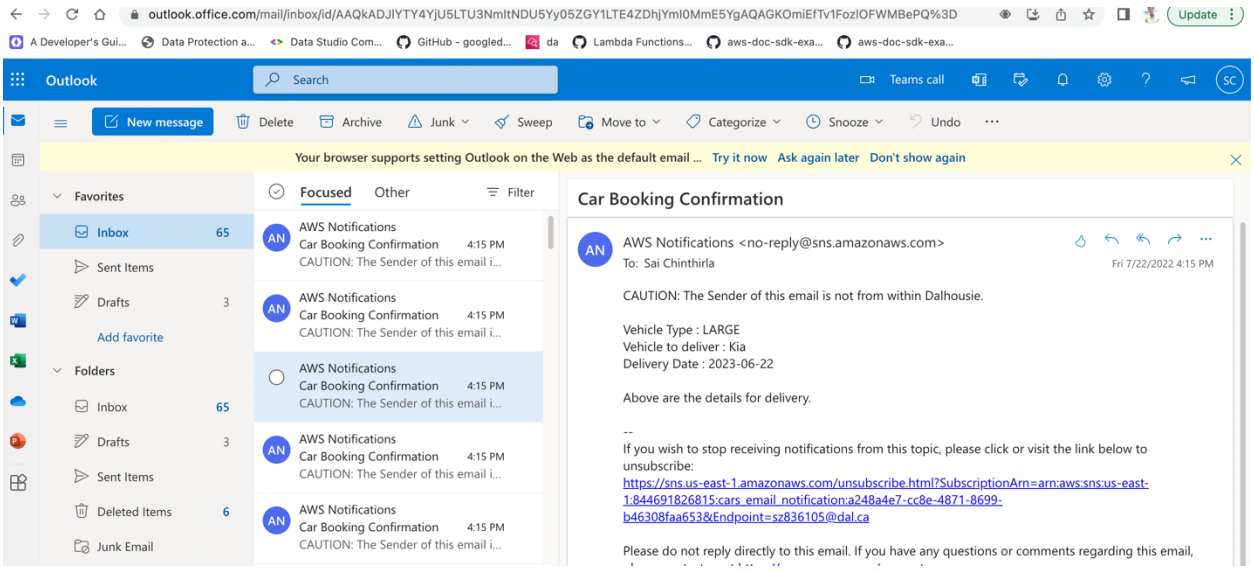
Screenshot of First email triggered:



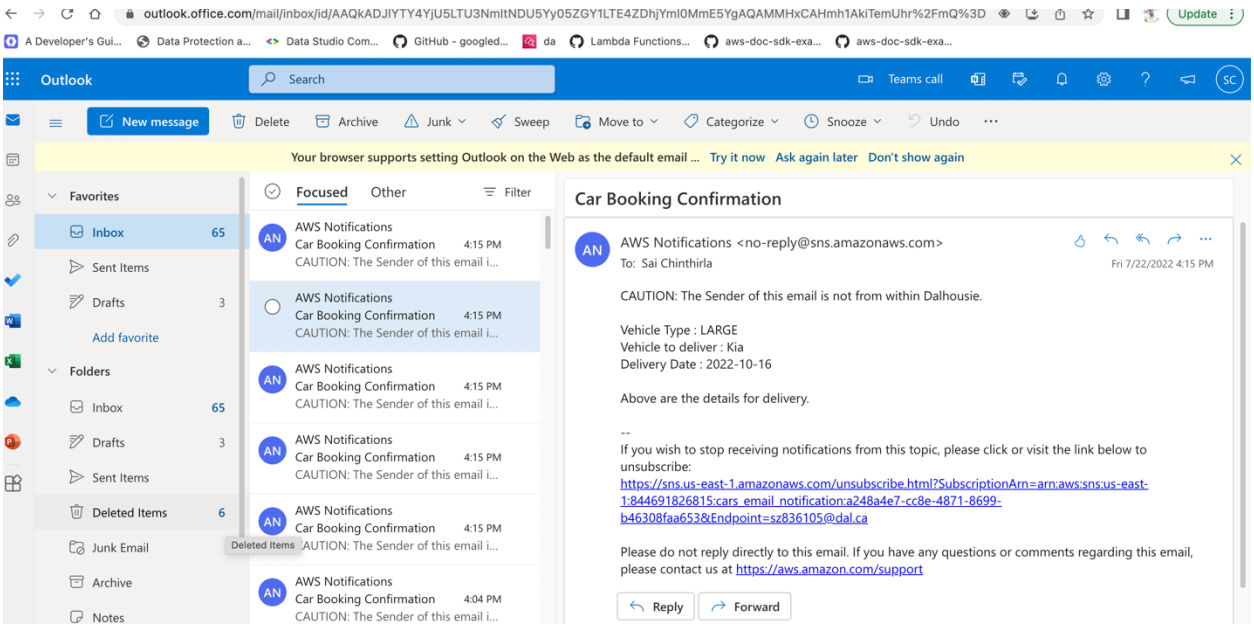
Screenshot of second email triggered:



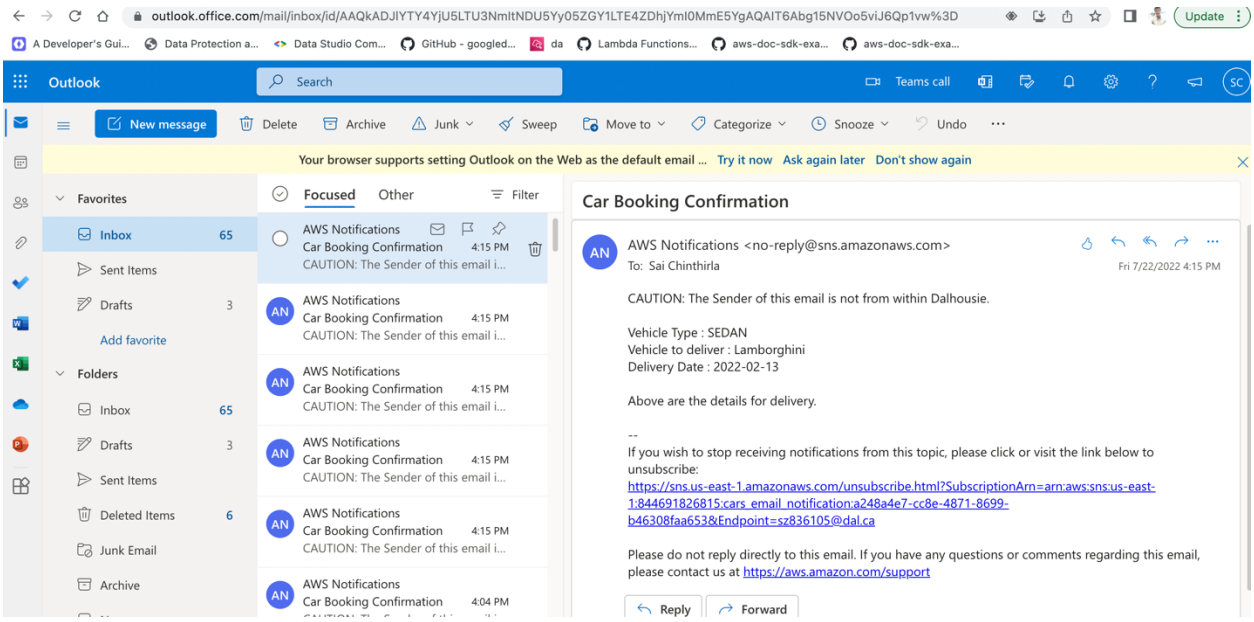
Screenshot of third email triggered:



Screenshot of fourth email triggered:



Screenshot of fifth email triggered:



GIT LAB LINK:

URL: https://git.cs.dal.ca/chinthirla/csci-5410-b00911631-saivikaschinthirla.git
Note: Code is present inside assignment – 5 folder.

References

[1] "Getting started with Amazon Sqs - Amazon Simple Queue Service." [Online]. Available: https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-getting-started.html. [Accessed: 22-Jul-2022].

[2] R. Treichler and C. Hardmeier, "Schlagwortnormdatei Schweiz für Allgemeine öffentliche Bibliotheken: SNS," *Amazon*, 2005. [Online]. Available: https://docs.aws.amazon.com/sns/latest/dg/welcome.html. [Accessed: 22-Jul-2022].