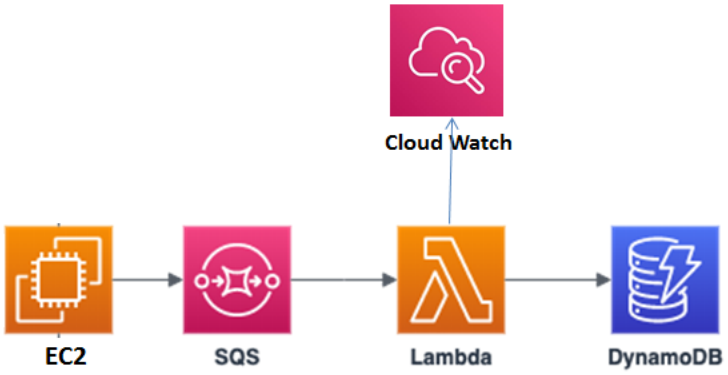
Open Ended Assignment

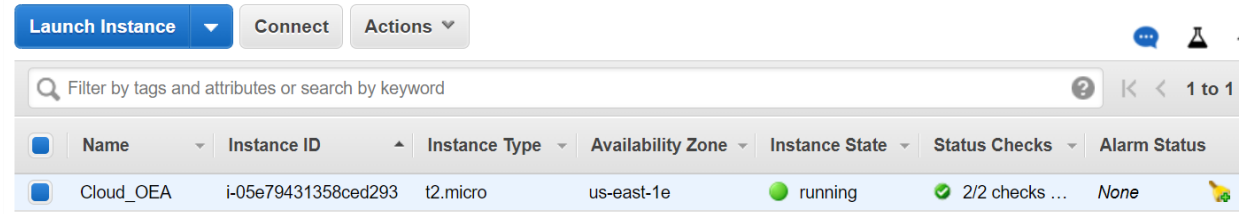
	Name	Roll No	Seat no.
1.	Pranav Hatwar	6	T214132

Problem Statement:- Design an web application for MIT-AOE Mess to collect data of Students and Store it in Dynamodb using SQS ,Lambda Function and EC2.

Architectures with AWS Lambda, SQS, DynamoDB, Cloud Watch and EC2



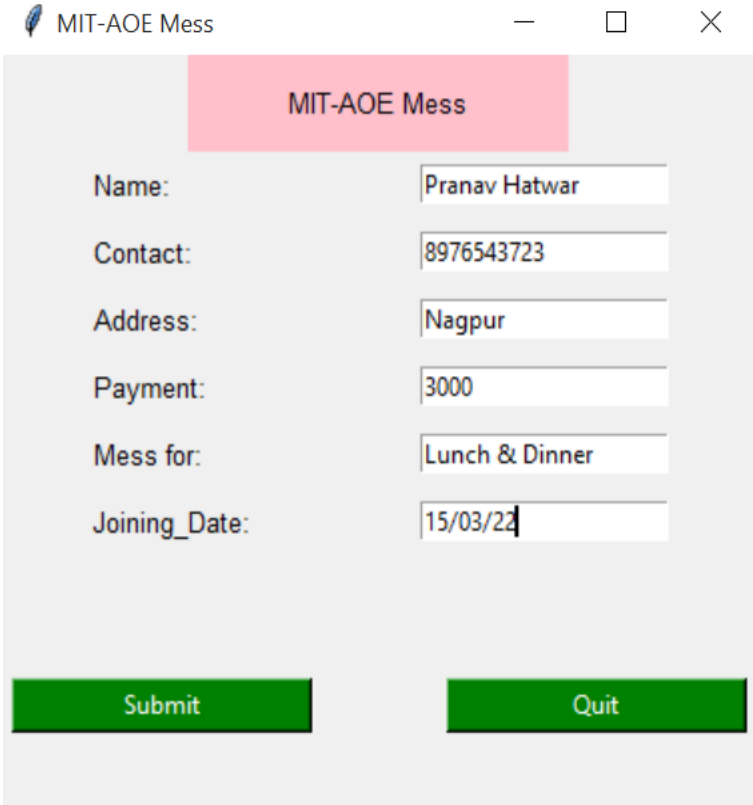
1. EC2 Instance




```
       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-58-166 ~]$ sudo su
[root@ip-172-31-58-166 ec2-user]# ls
CoudOEA
[root@ip-172-31-58-166 ec2-user]# cd CoudOEA
[root@ip-172-31-58-166 CoudOEA]# ls
SqS_Dyna.py
[root@ip-172-31-58-166 CoudOEA]# Python SqS_Dyna.py
```

2. Application

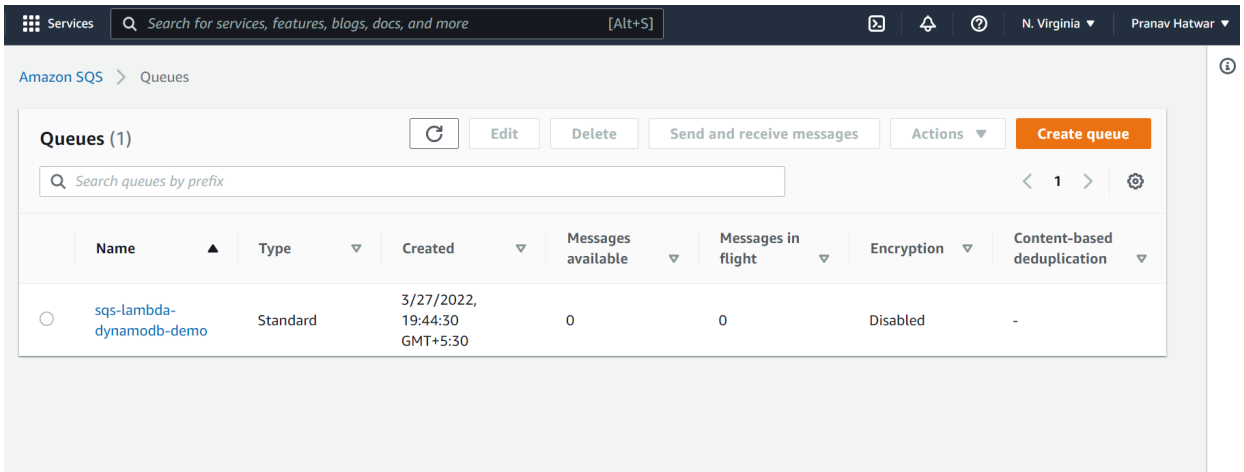


3. Message -Id

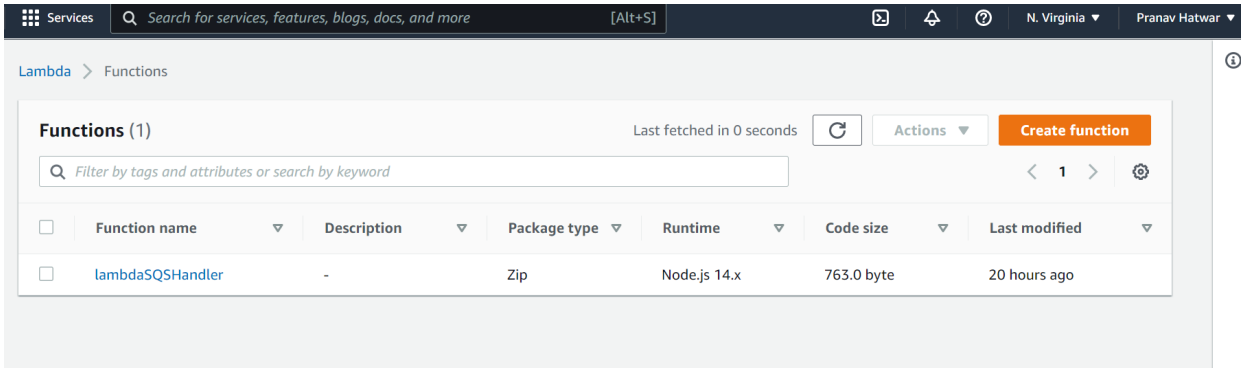
```
ace0c2e3-0a5a-4fb1-b302-b715f4f6e906

Process finished with exit code 0
```

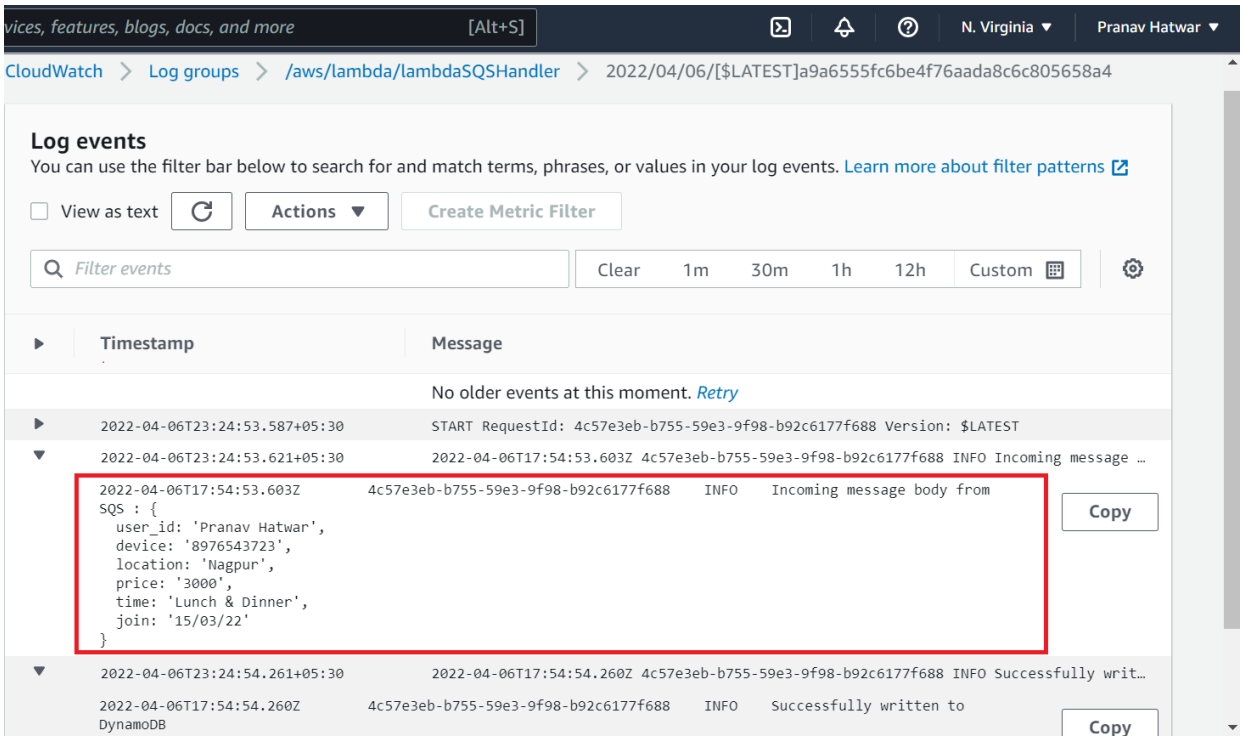
4. SQS



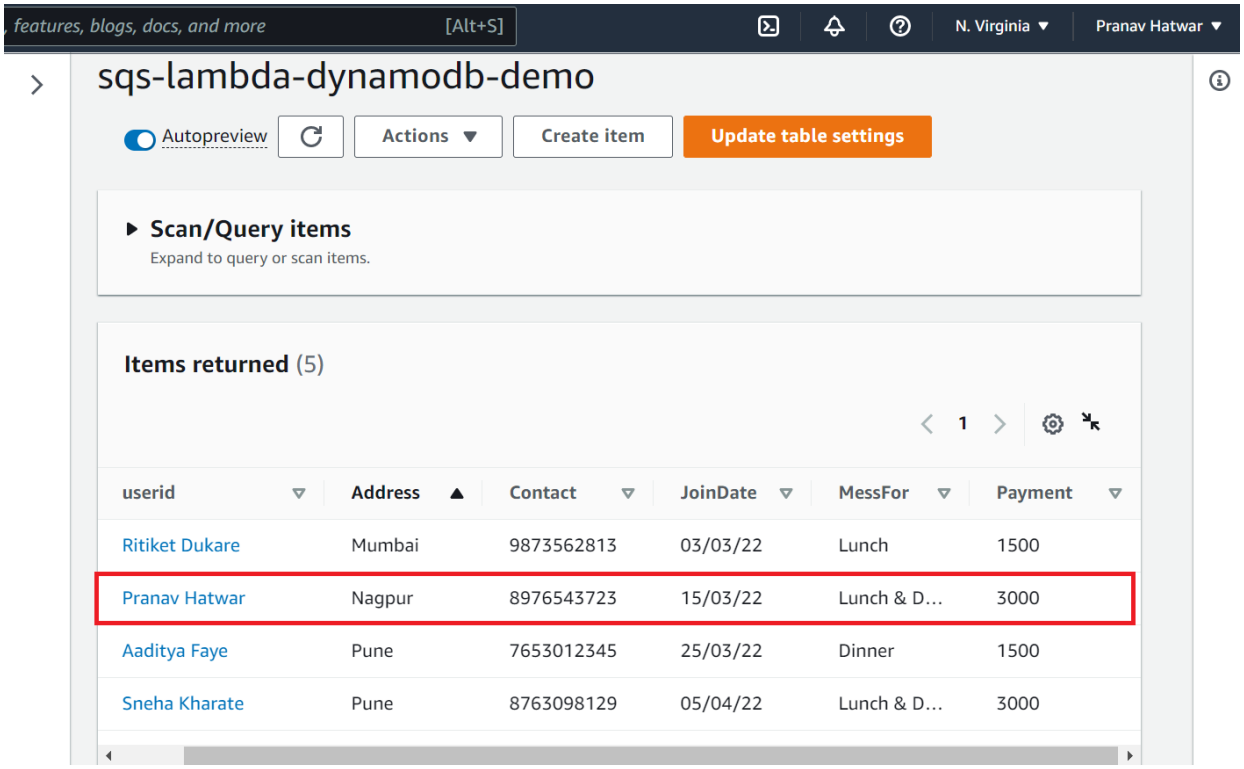
5. Lambda Function



6. Cloud Watch



7. Dynamodb Table



● Python Code

```
import tkinter
 import boto3

 from tkinter import *

 root = Tk()
 root.geometry("400x300")
 root.title("MIT-AOE Mess")
 var = StringVar()
 label = Label(root, textvariable=var, bg="pink", width=20, font=("bold", 10),bd=5,
 justify=RIGHT, padx=10, pady=10)
 var.set("MIT-AOE Mess")
 label.pack()
 fields = ('Name', 'Contact', 'Address', 'Payment', 'Mess for','Joining_Date')

 def monthly_payment(entries ):
    # variable
    global a, b, c, d, f ,k
    a = str(entries['Name'].get())
    b = str(entries['Contact'].get())
```

```
    c = str(entries['Address'].get())
    d = str(entries['Payment'].get())
    f = str(entries['Mess for'].get())
    k = str(entries['Joining_Date'].get())

def makeform(root, fields):
    entries = {}
    for field in fields:
        row = Frame(root)
        lab = Label(row, width=20, font=("bold", 10),text=field + ": ", anchor='w')
        ent = Entry(row)
        ent.insert(0, "0")
        row.pack(side=TOP, padx=5, pady=5)
        lab.pack(side=LEFT)
        ent.pack(side=RIGHT, expand=YES)
        entries[field] = ent
    return entries


if __name__ == '__main__':
    ents = makeform(root, fields)
    root.bind('<Return>', (lambda event, e=ents: fetch(e)))

    b2 = Button(root, text='Submit', width=20, bg='green', fg='white',command=(lambda e=ents:
monthly_payment(e)))
    b2.pack(side=LEFT, padx=5, pady=5)
    b3 = Button(root, text='Quit', width=20, bg='green', fg='white', command=root.quit)
    b3.pack(side=RIGHT, padx=5, pady=5)
    root.mainloop()
# Create SQS client
sqs = boto3.client('sqs')

queue_url =
'https://sqs.us-east-1.amazonaws.com/278460603394/sqs-lambda-dynamodb-demo'

# Send message to SQS queue
response = sqs.send_message(
    QueueUrl=queue_url,
    DelaySeconds=10,
    MessageAttributes={
        'Title': {
            'DataType': 'String',
            'StringValue': 'The Whistler'
        },
```

```
        'Author': {
            'DataType': 'String',
            'StringValue': 'John Grisham'
        },
        'WeeksOn': {
            'DataType': 'Number',
            'StringValue': '6'
        }
    },
    MessageBody=(
            '{"user_id":"' + a + '","device":"' + b + '","location":"' + c + '","price":"' + d + '","time":"' +
f + '" ,"join":"' + k + '"}'
    )
)

print(response['MessageId'])
```

● Lambda Function

```
const AWS = require('aws-sdk');
const dynamoDB = new AWS.DynamoDB.DocumentClient({
    region:'us-east-1',
    apiVersion:'2012-08-10'
});

exports.handler = async (event) => {
  try{
      //messages coming in from SQS
      const {Records} = event;
      const body = JSON.parse(Records[0].body); // in this case, only one item is present in the
Records array
      console.log("Incoming message body from SQS :", body);

      // writing data to dynamo DB:
      const params = {
          TableName:'sqs-lambda-dynamodb-demo',
          Item:{
              userid : body.user_id,
              Contact : body.device,
              Address : body.location,
              Payment : body.price,
              MessFor : body.time,
              JoinDate : body.join
```

```
        }
    };

    //write data to dynamo DB:
    await dynamoDB.put(params).promise();

    //success logging to cloudwatch:
    console.log('Successfully written to DynamoDB');
  }catch(error){
    //error handling
    console.error('Error in executing lambda handler for SQS',error);
    return;
  }
};
```