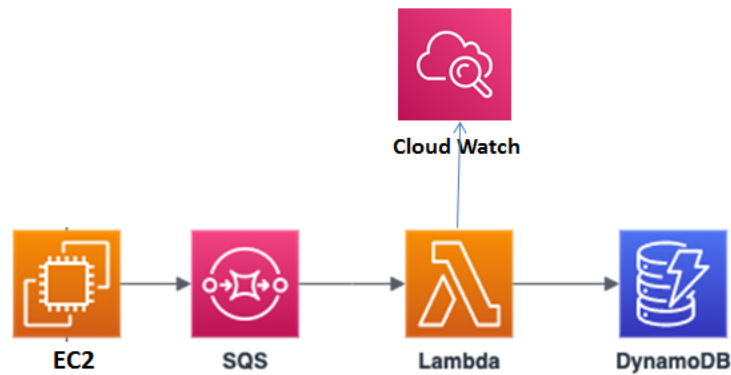


Open Ended Assignment

	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

Launch Instance

▼

Connect

Actions ▼

...

🔍

Filter by tags and attributes or search by keyword

?

⏪ ⏩ 1 to 1

<input type="checkbox"/>	Name ▼	Instance ID ▲	Instance Type ▼	Availability Zone ▼	Instance State ▼	Status Checks ▼	Alarm Status
<input type="checkbox"/>	Cloud_OEA	i-05e79431358ced293	t2.micro	us-east-1e	● running	✔ 2/2 checks ...	None

```
  _ |  _ |  _ |
 _ | ( _ |  _ |
 _ | \ _ |  _ |
      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
```

```
Coud0EA
```

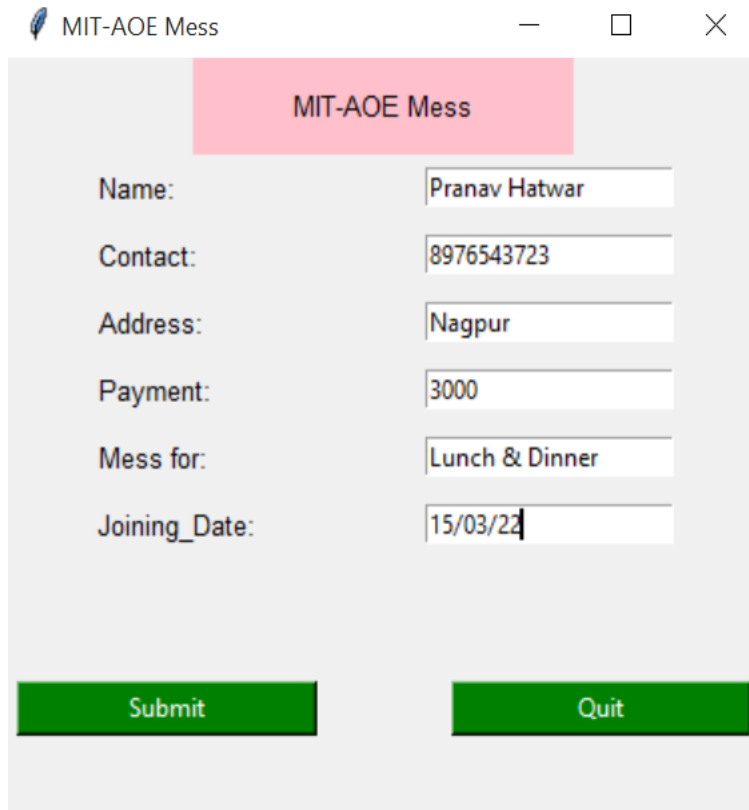
```
[root@ip-172-31-58-166 ec2-user]# cd Coud0EA
```

```
[root@ip-172-31-58-166 Coud0EA]# ls
```

```
SqS_Dyna.py
```

```
[root@ip-172-31-58-166 Coud0EA]# Python SqS_Dyna.py
```

2. Application



A screenshot of a desktop application window titled "MIT-AOE Mess". The window has a pink header bar with the title. Below the header, there are seven input fields with labels: "Name:", "Contact:", "Address:", "Payment:", "Mess for:", and "Joining_Date:". Each field contains text: "Pranav Hatwar", "8976543723", "Nagpur", "3000", "Lunch & Dinner", and "15/03/22" respectively. At the bottom of the window, there are two green buttons: "Submit" and "Quit".

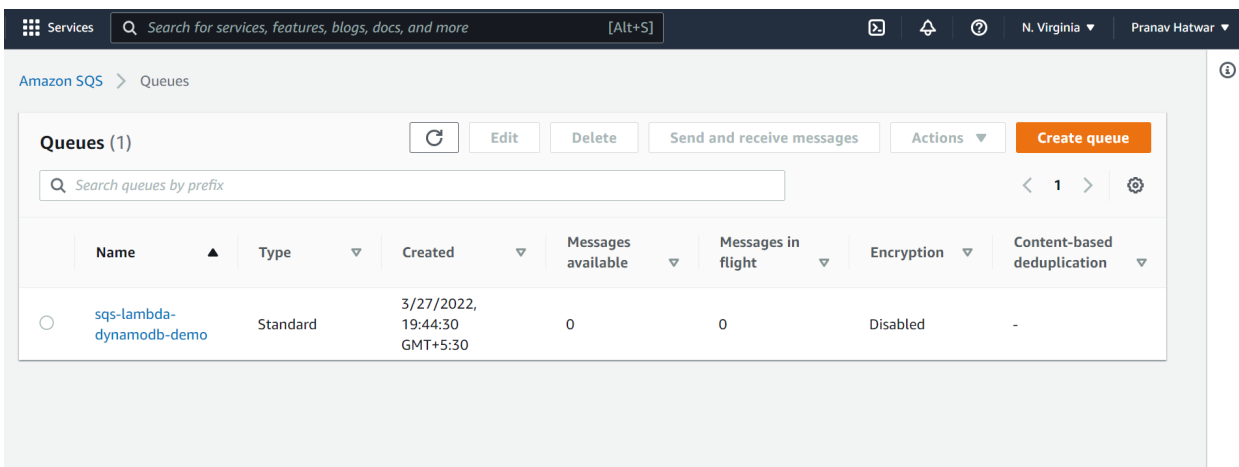
Field	Value
Name:	Pranav Hatwar
Contact:	8976543723
Address:	Nagpur
Payment:	3000
Mess for:	Lunch & Dinner
Joining_Date:	15/03/22

3. Message -Id

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

```
Process finished with exit code 0
```

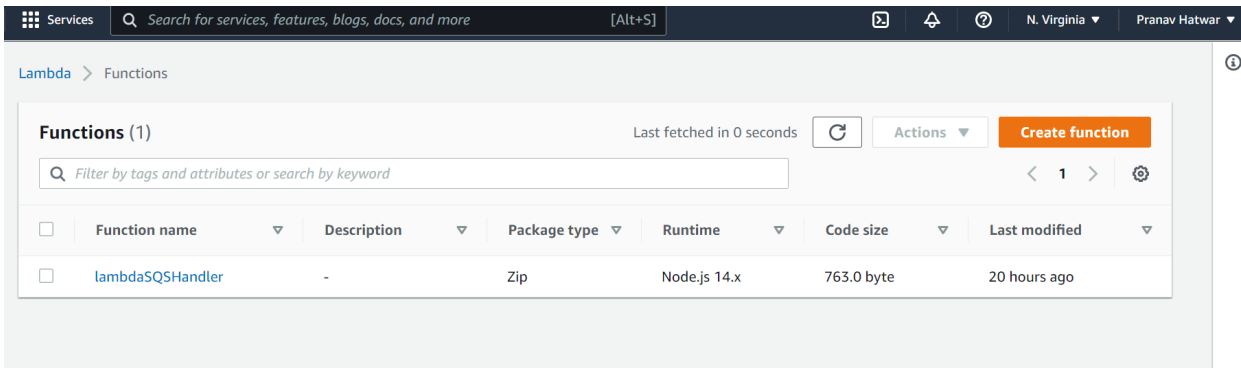
4. SQS



A screenshot of the Amazon SQS console. The top navigation bar shows "Services" and a search bar. The main content area is titled "Amazon SQS > Queues". It features a table of queues with columns: Name, Type, Created, Messages available, Messages in flight, Encryption, and Content-based deduplication. There is one queue listed: "sqs-lambda-dynamodb-demo" of type "Standard", created on "3/27/2022, 19:44:30 GMT+5:30", with 0 messages available and 0 in flight. Encryption is "Disabled" and deduplication is "-". Above the table, there are buttons for "Edit", "Delete", "Send and receive messages", "Actions", and "Create queue". A search bar for queues is also present.

Name	Type	Created	Messages available	Messages in flight	Encryption	Content-based deduplication
sqs-lambda-dynamodb-demo	Standard	3/27/2022, 19:44:30 GMT+5:30	0	0	Disabled	-

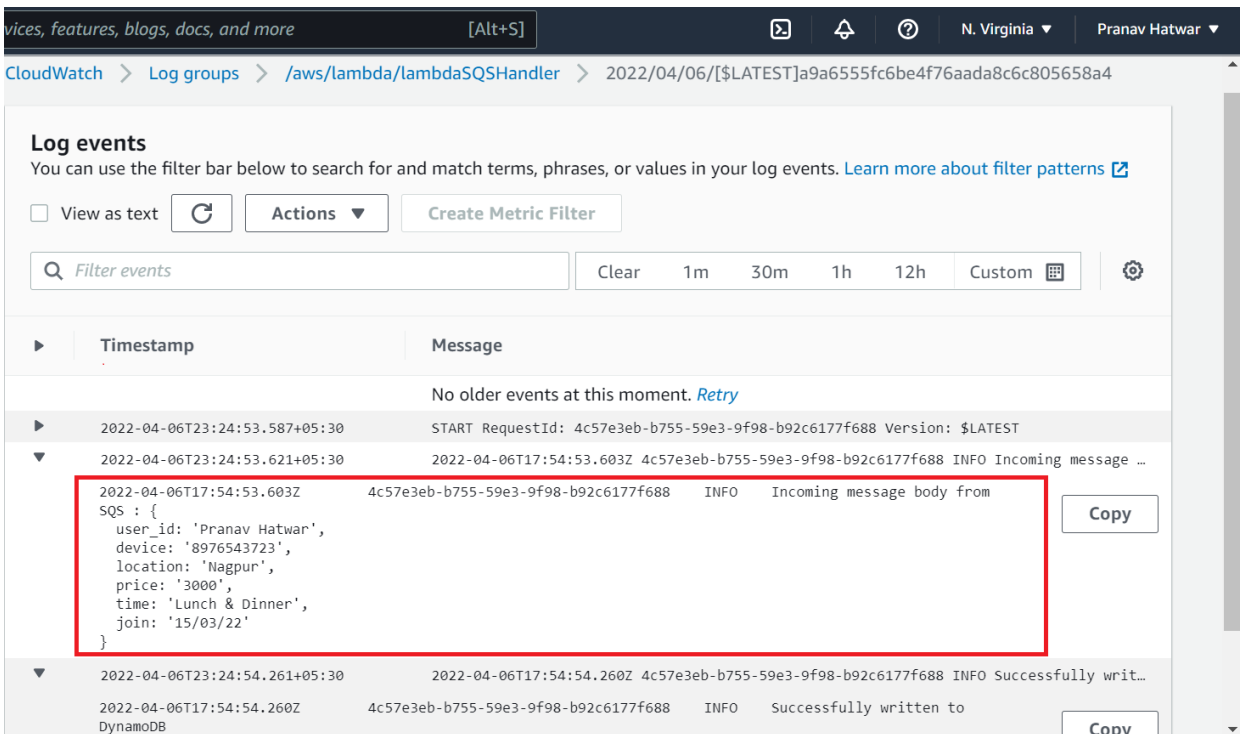
5. Lambda Function



The screenshot shows the AWS Lambda console. At the top, there's a search bar and navigation links. Below, the 'Functions (1)' section displays a table with one function: 'lambdaSQSHandler'. The table columns are: Function name, Description, Package type, Runtime, Code size, and Last modified. The function 'lambdaSQSHandler' has a description of '-', package type of 'Zip', runtime of 'Node.js 14.x', code size of '763.0 byte', and was last modified '20 hours ago'. There are buttons for 'Create function' and 'Actions'.

Function name	Description	Package type	Runtime	Code size	Last modified
lambdaSQSHandler	-	Zip	Node.js 14.x	763.0 byte	20 hours ago




6. Cloud Watch





The screenshot shows the AWS CloudWatch console, specifically the 'Log events' section for the Lambda function 'lambdaSQSHandler'. The interface includes a search bar, filters, and a table of log events. The first event is highlighted with a red box, showing an incoming message body from an SQS queue. The message body is a JSON object containing user information.

Timestamp	Message
2022-04-06T17:54:53.603Z	START RequestId: 4c57e3eb-b755-59e3-9f98-b92c6177f688 Version: \$LATEST
2022-04-06T17:54:53.621+05:30	2022-04-06T17:54:53.603Z 4c57e3eb-b755-59e3-9f98-b92c6177f688 INFO Incoming message ...
2022-04-06T17:54:53.603Z	4c57e3eb-b755-59e3-9f98-b92c6177f688 INFO Incoming message body from SQS : { user_id: 'Pranav Hatwar', device: '8976543723', location: 'Nagpur', price: '3000', time: 'Lunch & Dinner', join: '15/03/22' }
2022-04-06T17:54:54.261+05:30	2022-04-06T17:54:54.260Z 4c57e3eb-b755-59e3-9f98-b92c6177f688 INFO Successfully writ...
2022-04-06T17:54:54.260Z	4c57e3eb-b755-59e3-9f98-b92c6177f688 INFO Successfully written to DynamoDB

7. Dynamodb Table



features, blogs, docs, and more [Alt+S]    N. Virginia ▾ Pranav Hatwar ▾

> sqs-lambda-dynamodb-demo ⓘ

 Autopreview  Actions ▾ Create item Update table settings

► **Scan/Query items**
Expand to query or scan items.

Items returned (5)

< 1 >  

userid ▾	Address ▲	Contact ▾	JoinDate ▾	MessFor ▾	Payment ▾
Ritiket Dukare	Mumbai	9873562813	03/03/22	Lunch	1500
Pranav Hatwar	Nagpur	8976543723	15/03/22	Lunch & D...	3000
Aaditya Faye	Pune	7653012345	25/03/22	Dinner	1500
Sneha Kharate	Pune	8763098129	05/04/22	Lunch & D...	3000

● 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;  
}  
};
```