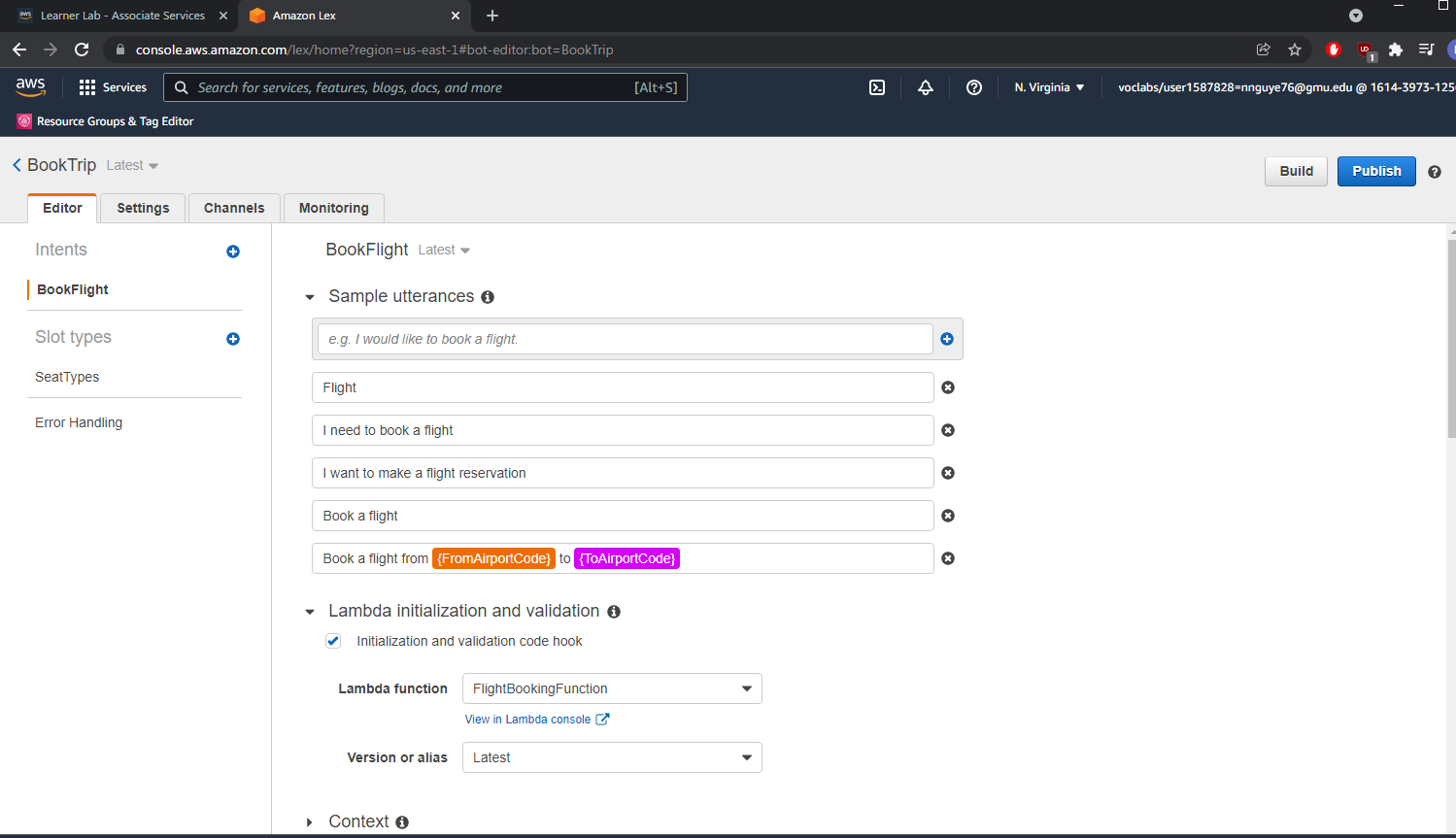
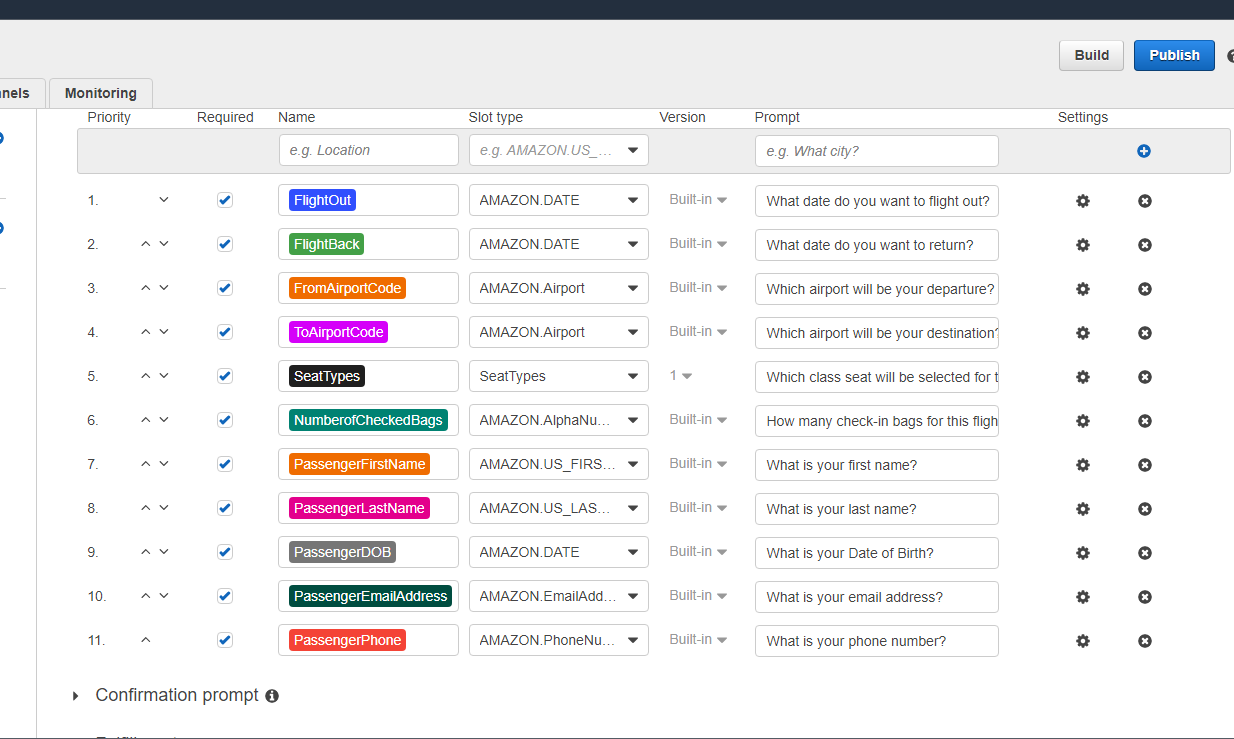
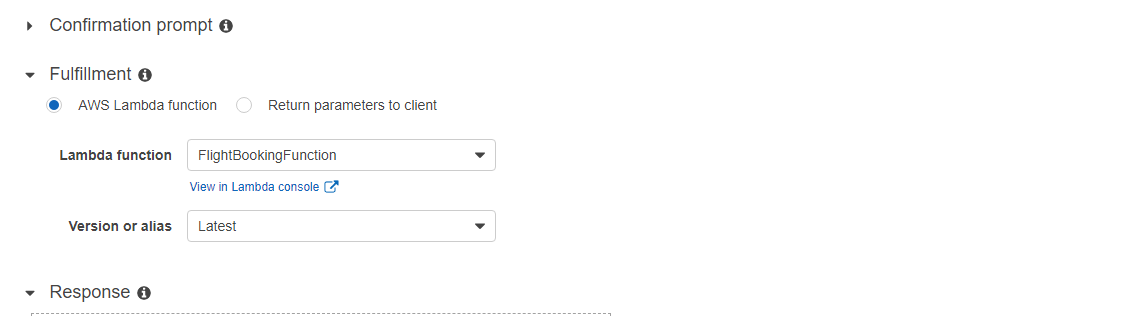
**Relevant screenshots of chatbot**

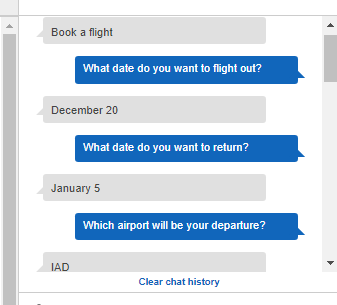
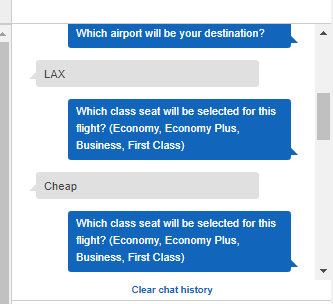
**Amazon LEX**

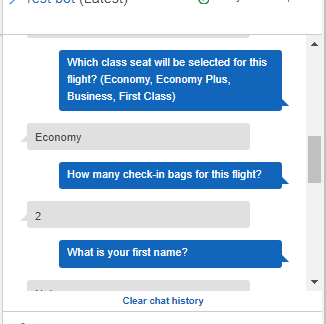
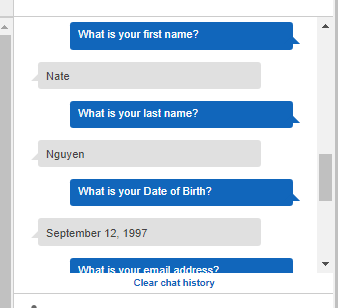


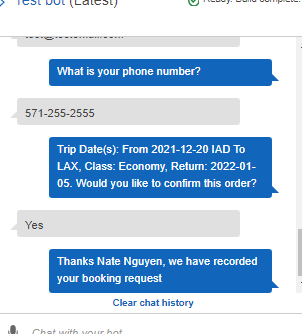
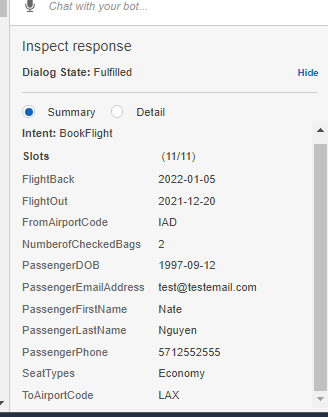




**Test Chatbot Sample:**

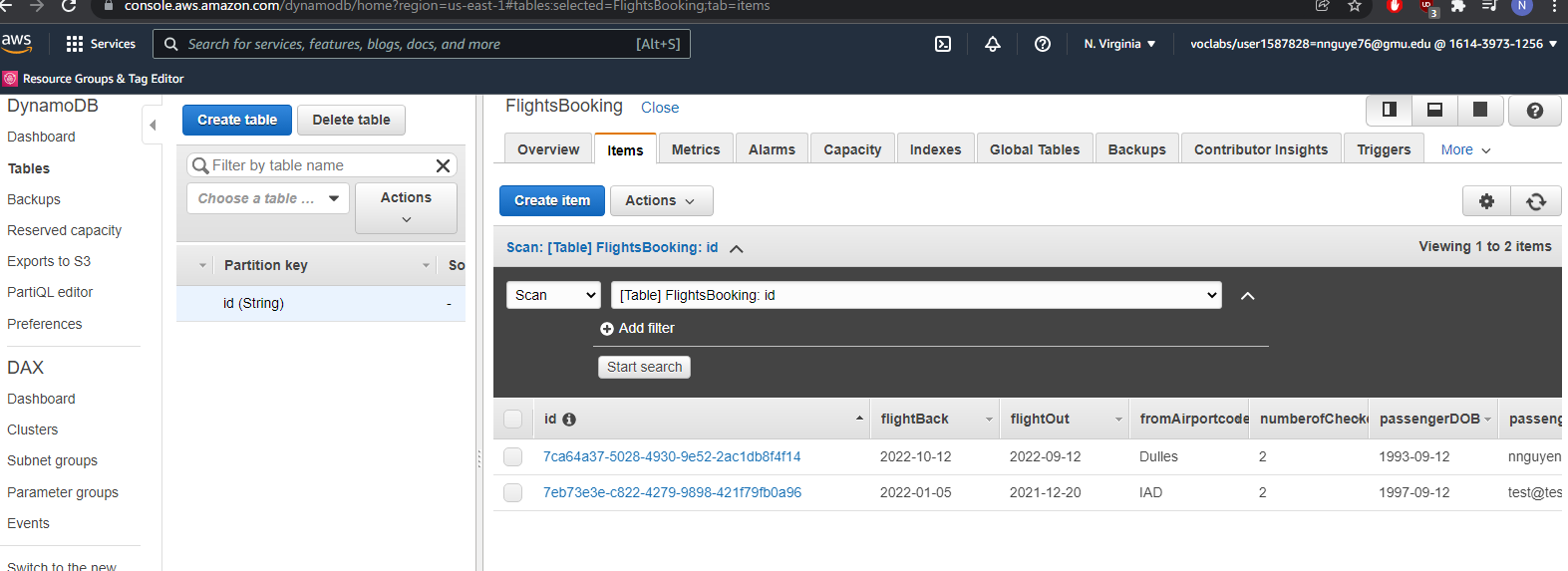
 

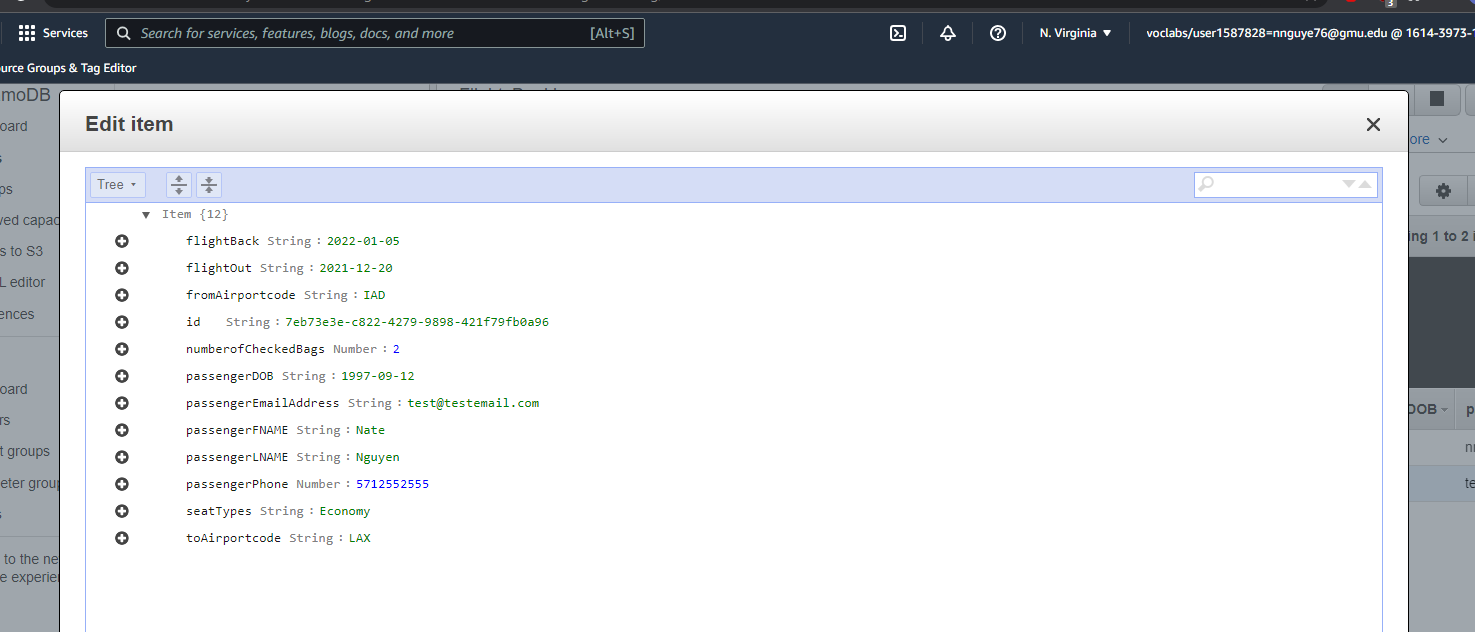
 

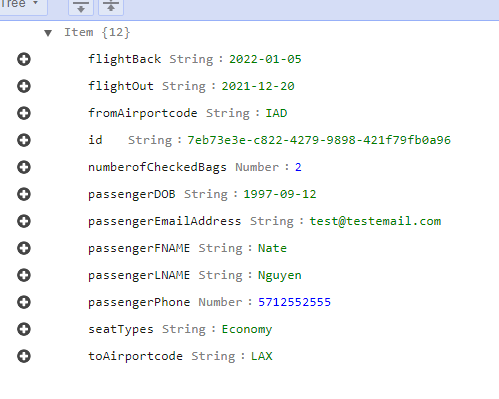
 

**DynamoDB table Content**

**Table:**

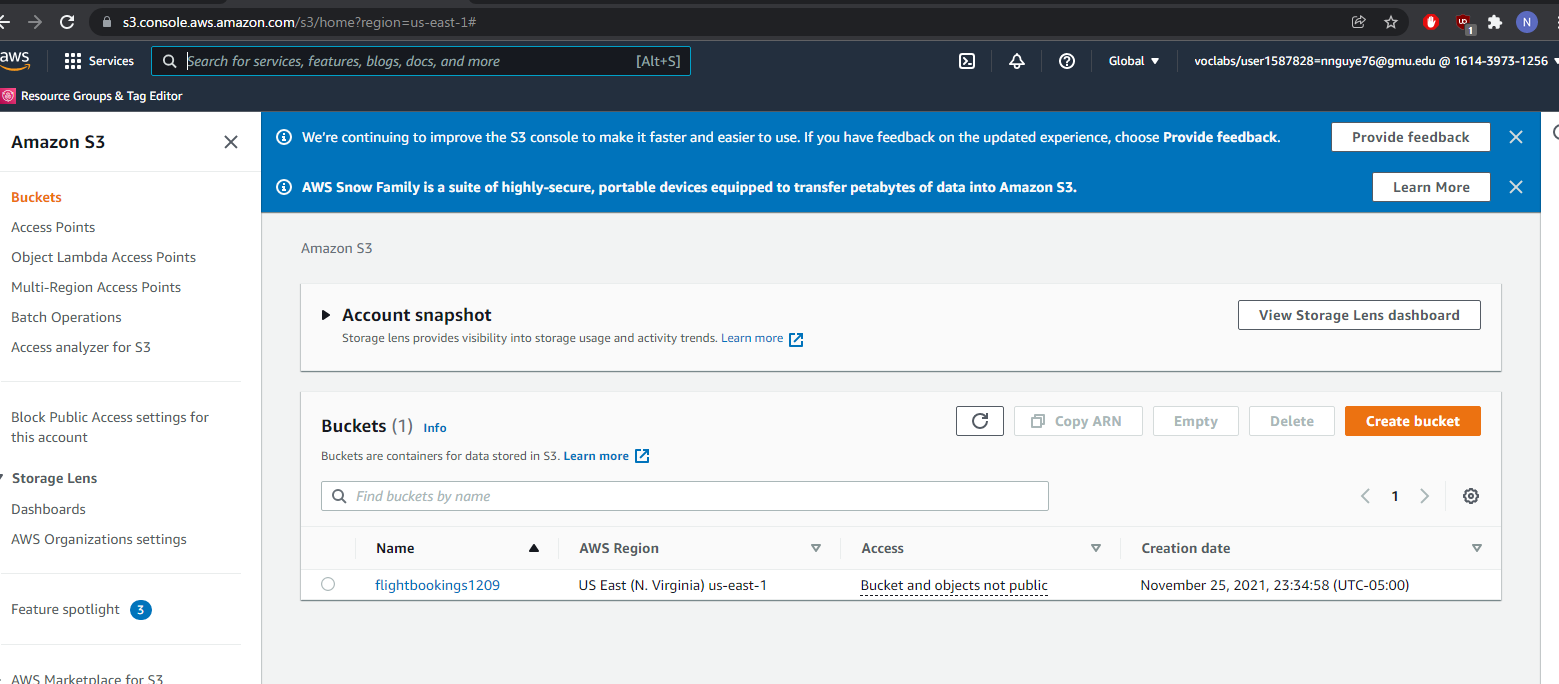


**Data:** 

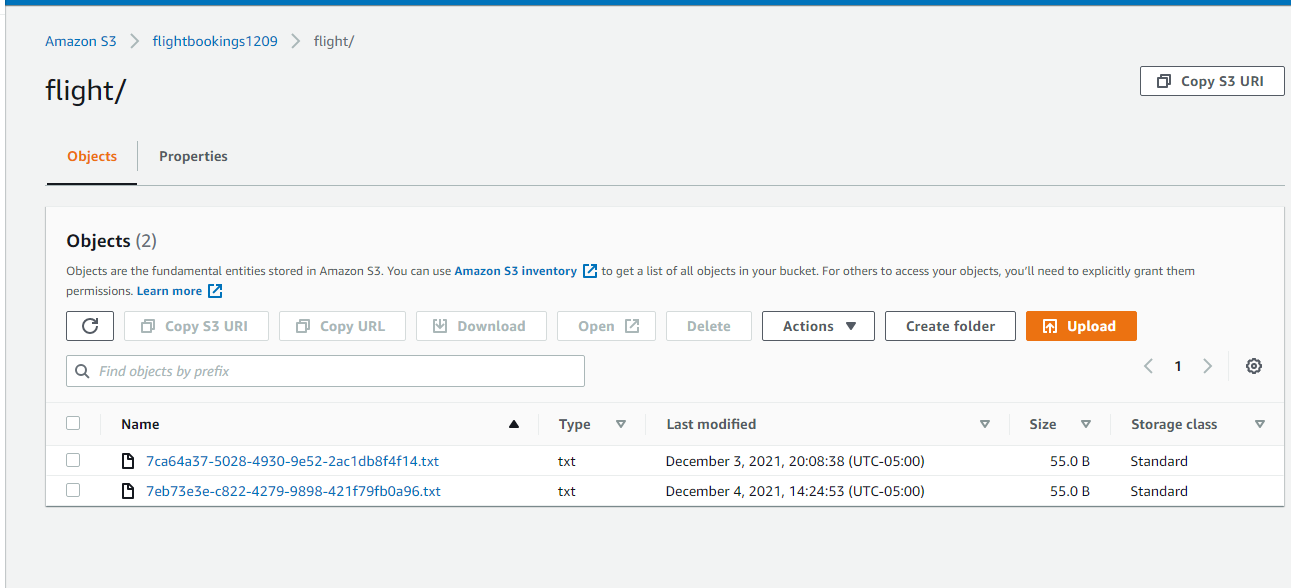


**List S3 Bucket**

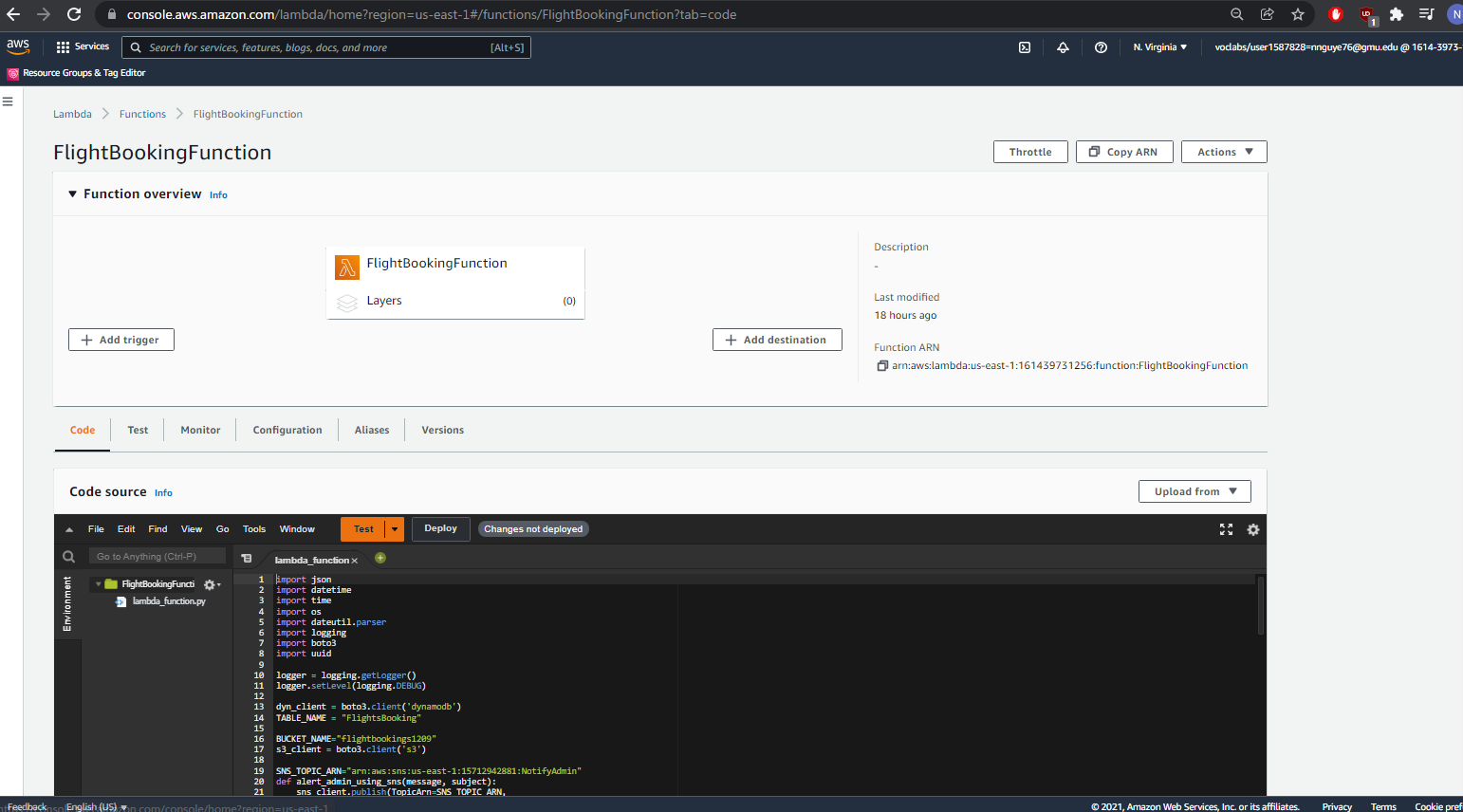
**Bucket:**



**Upload Text File to Bucket:**



**Lambda Function Code**



**Code:**

**import json**

**import datetime**

**import time**

**import os**

**import dateutil.parser**

**import logging**

**import boto3**

**import uuid**

**logger = logging.getLogger()**

**logger.setLevel(logging.DEBUG)**

**dyn\_client = boto3.client('dynamodb')**

**TABLE\_NAME = "FlightsBooking"**

**BUCKET\_NAME="flightbookings1209"**

**s3\_client = boto3.client('s3')**

**SNS\_TOPIC\_ARN="arn:aws:sns:us-east-1:15712942881:NotifyAdmin"**

**def alert\_admin\_using\_sns(message, subject):**

**sns\_client.publish(TopicArn=SNS\_TOPIC\_ARN,**

**Message=message,**

**Subject=subject)**

**def store\_details\_to\_s3(bucket, key, data):**

**s3\_client.put\_object(Body=data, Bucket=bucket, Key=key)**

**def dispatch(intent\_request):**

**logger.debug('dispatch userId={}, intentName={}'.format(intent\_request['userId'], intent\_request['currentIntent']['name']))**

**intent\_name = intent\_request['currentIntent']['name']**

**if intent\_name == 'BookFlight':**

**return take\_flightbooking(intent\_request)**

**raise Exception('Intent with name ' + intent\_name + ' not supported')**

**def lambda\_handler(event, context):**

**os.environ['TZ'] = 'America/New\_York'**

**time.tzset()**

**logger.debug('event.bot.name={}'.format(event['bot']['name']))**

**return dispatch(event)**

**# --- Helper Functions ---**

**def safe\_int(n):**

**"""**

**Safely convert n value to int.**

**"""**

**if n is not None:**

**return int(n)**

**return n**

**def try\_ex(func):**

**"""**

**Call passed in function in try block. If KeyError is encountered return None.**

**This function is intended to be used to safely access dictionary.**

**Note that this function would have negative impact on performance.**

**"""**

**try:**

**return func()**

**except KeyError:**

**return None**

**def elicit\_slot(session\_attributes, intent\_name, slots, slot\_to\_elicit, message):**

**return {**

**'sessionAttributes': session\_attributes,**

**'dialogAction': {**

**'type': 'ElicitSlot',**

**'intentName': intent\_name,**

**'slots': slots,**

**'slotToElicit': slot\_to\_elicit,**

**'message': message**

**}**

**}**

**def confirm\_intent(session\_attributes, intent\_name, slots, message):**

**return {**

**'sessionAttributes': session\_attributes,**

**'dialogAction': {**

**'type': 'ConfirmIntent',**

**'intentName': intent\_name,**

**'slots': slots,**

**'message': message**

**}**

**}**

**def close(session\_attributes, fulfillment\_state, message):**

**response = {**

**'sessionAttributes': session\_attributes,**

**'dialogAction': {**

**'type': 'Close',**

**'fulfillmentState': fulfillment\_state,**

**'message': message**

**}**

**}**

**return response**

**def delegate(session\_attributes, slots):**

**return {**

**'sessionAttributes': session\_attributes,**

**'dialogAction': {**

**'type': 'Delegate',**

**'slots': slots**

**}**

**}**

**#-- DynamoDB Save**

**def save\_flight(flightOut, flightBack, fromAirportcode, toAirportcode, seatTypes, numberofCheckedBags, passengerFNAME, passengerLNAME, passengerDOB, passengerEmailAddress, passengerPhone):**

**id = str(uuid.uuid4())**

**flightBookings = json.dumps({**

**'flightOut': flightOut,**

**'flightBack': flightBack,**

**'fromAirportcode': fromAirportcode,**

**'toAirportcode': toAirportcode,**

**'seatTypes': seatTypes,**

**'numberofCheckedBags': numberofCheckedBags,**

**'passengerFNAME': passengerFNAME,**

**'passengerLNAME': passengerLNAME,**

**'passengerDOB': passengerDOB,**

**'passengerEmailAddress': passengerEmailAddress,**

**'passengerPhone': str(passengerPhone),**

**})**

**data = dyn\_client.put\_item(**

**TableName=TABLE\_NAME,**

**Item={**

**'id': {**

**'S': id**

**},**

**'flightOut': {**

**'S': flightOut**

**},**

**'flightBack': {**

**'S': flightBack**

**},**

**'fromAirportcode': {**

**'S': fromAirportcode**

**},**

**'toAirportcode': {**

**'S': toAirportcode**

**},**

**'seatTypes': {**

**'S': seatTypes**

**},**

**'numberofCheckedBags': {**

**'N': numberofCheckedBags**

**},**

**'passengerFNAME': {**

**'S': passengerFNAME**

**},**

**'passengerLNAME': {**

**'S': passengerLNAME**

**},**

**'passengerDOB': {**

**'S': passengerDOB**

**},**

**'passengerEmailAddress': {**

**'S': passengerEmailAddress**

**},**

**'passengerPhone': {**

**'N': str(passengerPhone)**

**}**

**}**

**)**

**store\_details\_to\_s3(BUCKET\_NAME, "flight/{}.txt".format(id), flightBookings)**

**# --- Validation Functions ---**

**def isvalid\_seatType(seatType):**

**seatTypes = ['economy', 'economy plus', 'business', 'first class']**

**return seatType.lower() in seatTypes**

**def validate\_bookings(slots):**

**flightOut = try\_ex(lambda: slots['FlightOut'])**

**flightBack = try\_ex(lambda: slots['FlightBack'])**

**fromAirportcode = try\_ex(lambda: slots['FromAirportCode'])**

**toAirportcode = try\_ex(lambda: slots['ToAirportCode'])**

**seatType = try\_ex(lambda: slots['SeatTypes'])**

**numberofCheckedBags = safe\_int(try\_ex(lambda: slots['NumberCheckedBags']))**

**passengerFNAME = try\_ex(lambda: slots['PassengerFirstName'])**

**passengerLNAME = try\_ex(lambda: slots['PassengerLastName'])**

**passengerDOB = try\_ex(lambda: slots['PassengerDOB'])**

**passengerEmailAddress = try\_ex(lambda: slots['PassengerEmailAddress'])**

**passengerPhone = try\_ex(lambda: slots['PassengerPhone'])**

**if seatType and not isvalid\_seatType(seatType):**

**return build\_validation\_result(**

**False,**

**'SeatTypes',**

**'I did not recognize that Class Seat type. Which class seat will be selected for this flight? (Economy, Economy Plus, Business, First) '**

**)**

**return {'isValid': True}**

**def build\_validation\_result(isvalid, violated\_slot, message\_content):**

**return {**

**'isValid': isvalid,**

**'violatedSlot': violated\_slot,**

**'message': {'contentType': 'PlainText', 'content': message\_content}**

**}**

**def take\_flightbooking(intent\_request):**

**slots = intent\_request['currentIntent']['slots']**

**flightOut = slots['FlightOut']**

**flightBack = slots['FlightBack']**

**fromAirportcode = slots['FromAirportCode']**

**toAirportcode = slots['ToAirportCode']**

**seatType = slots['SeatTypes']**

**numberofCheckedBags = slots['NumberofCheckedBags']**

**passengerFNAME = slots['PassengerFirstName']**

**passengerLNAME = slots['PassengerLastName']**

**passengerDOB = slots['PassengerDOB']**

**passengerEmailAddress = slots['PassengerEmailAddress']**

**passengerPhone = slots['PassengerPhone']**

**session\_attributes = intent\_request['sessionAttributes'] if intent\_request['sessionAttributes'] is not None else {}**

**logger.debug(intent\_request['invocationSource'])**

**if intent\_request['invocationSource'] == 'DialogCodeHook':**

**validation\_result = validate\_bookings(intent\_request['currentIntent']['slots'])**

**logger.debug(validation\_result)**

**if not validation\_result['isValid']:**

**slots[validation\_result['violatedSlot']] = None**

**return elicit\_slot(**

**session\_attributes,**

**intent\_request['currentIntent']['name'],**

**slots,**

**validation\_result['violatedSlot'],**

**validation\_result['message']**

**)**

**return delegate(session\_attributes, intent\_request['currentIntent']['slots'])**

**save\_flight(flightOut, flightBack, fromAirportcode, toAirportcode, seatType, numberofCheckedBags, passengerFNAME, passengerLNAME, passengerDOB, passengerEmailAddress, passengerPhone)**

**return close(**

**session\_attributes,**

**'Fulfilled',**

**{**

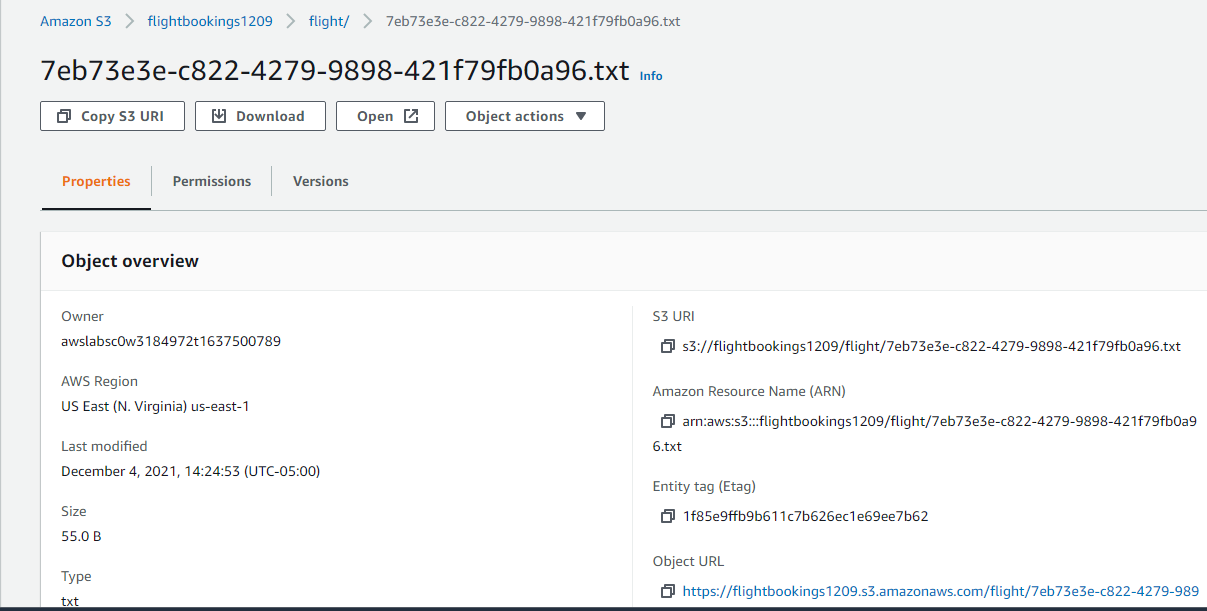
**'contentType': 'PlainText',**

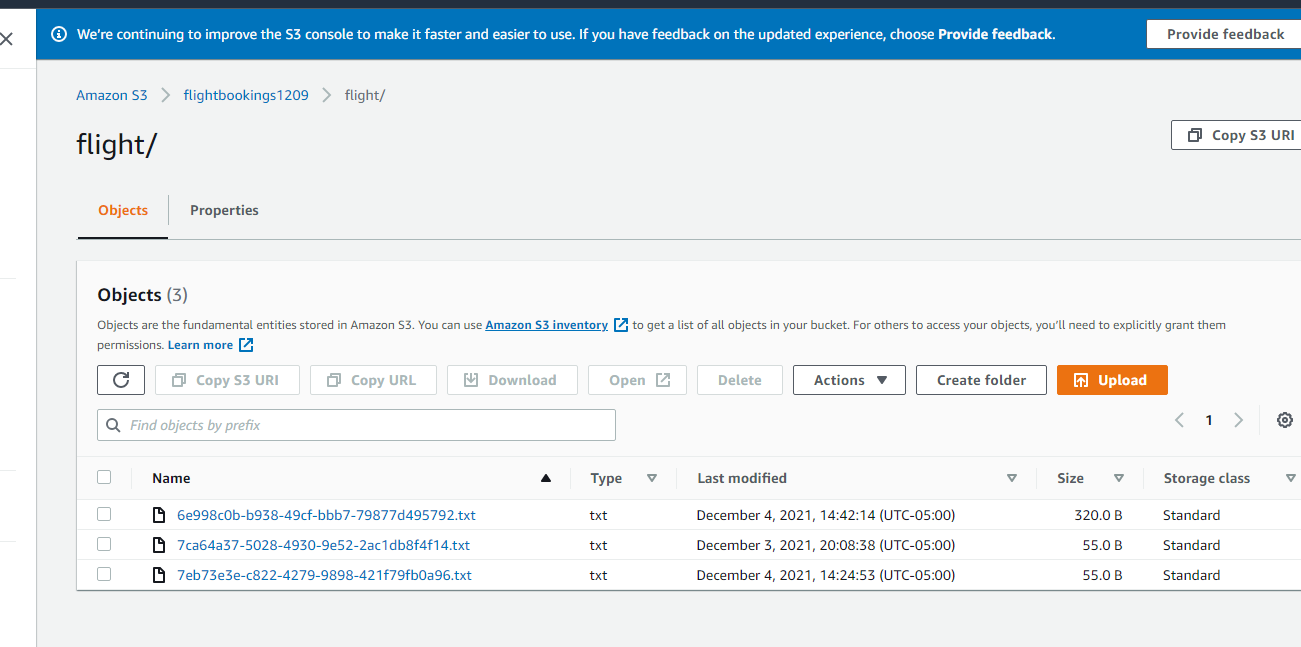
**'content': 'Thanks {} {}, we have recorded your booking request'.format(passengerFNAME, passengerLNAME)**

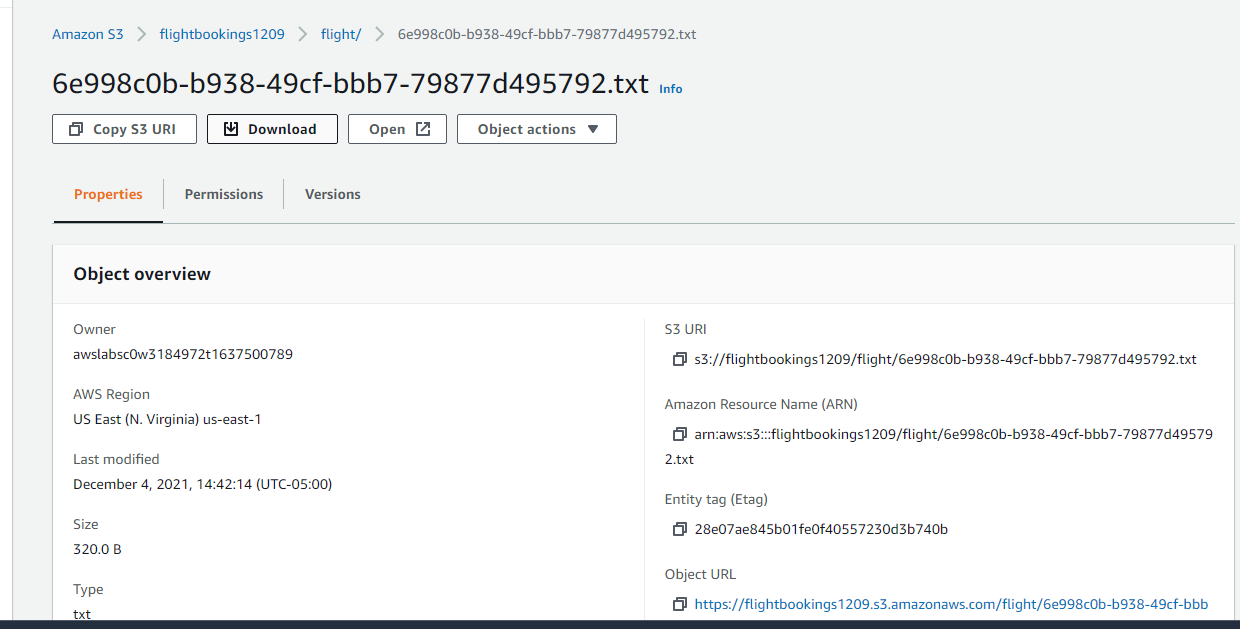
**}**

**)**

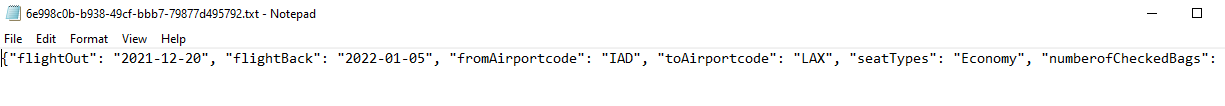
**Trip Text File**

**File:**





**TXT File:**



**{"flightOut": "2021-12-20", "flightBack": "2022-01-05", "fromAirportcode": "IAD", "toAirportcode": "LAX", "seatTypes": "Economy", "numberofCheckedBags": "2", "passengerFNAME": "Nate", "passengerLNAME": "Nguyen", "passengerDOB": "1997-09-12", "passengerEmailAddress": "test@testemail.com", "passengerPhone": "5712552555"}**