import json  
import logging  
import boto3  
from botocore.exceptions import ClientError  
  
finalOutput = {}  
s3\_resource = boto3.resource("s3")  
s3\_client = boto3.client('s3')  
  
  
def lambda\_handler():  
 labeling\_data = get\_output\_manifest()  
 process\_labels(labeling\_data)  
 return {  
 'statusCode': 200,  
 'body': labeling\_data  
 }  
  
  
# To get manifest from S3 to process further  
def get\_output\_manifest():  
 bucket\_name = 'output-manifest'  
 objects = list\_bucket\_objects(bucket\_name)  
 output\_manifest = []  
 if objects is not None:  
 for obj in objects:  
 try:  
 data = s3\_client.get\_object(Bucket=bucket\_name, Key=obj["Key"])  
 contents = data['Body'].read()  
 file\_content = contents.decode("utf-8")  
 json\_content = json.loads(file\_content)  
 output\_manifest.append(json\_content)  
 except ClientError as e:  
 logging.error(e)  
 return output\_manifest  
  
  
# publish final combined output manifest to S3  
def write\_final\_output\_tos3(bucket\_name, data):  
 s3\_resource.Object(bucket\_name, 'combined-output-manifest.json').put(Body=json.dumps(data))  
  
  
def final\_labeling\_job\_result(result, labeler\_selections, sentence):  
 labeler\_selections.append({"majorityAnswer": result[0]})  
 finalOutput[sentence] = labeler\_selections  
  
  
# build task for tie  
def build\_task(sentence, idx):  
 task = {  
 "id": idx + 1,  
 "metadata": {  
 "audioFileUrl": "./assets/french-fuse.mp3",  
 "audioText": sentence  
 },  
 "labelerResponse": [  
  
 ]  
 }  
 return task  
  
  
# create tie breaker job and push it to S3 bucket  
def create\_tie\_breaker\_job(data):  
 tie\_breaker\_jobs = {}  
  
 # 2 - Tie break Task  
 tie\_breaker\_job = {  
 "id": 1,  
 "type": 2,  
 "tasks": [],  
 "workedBy": [  
  
 ]  
 }  
 tie\_tasks = []  
 for idx, v in enumerate(data):  
 tie\_tasks.append(build\_task(v, idx))  
 tie\_breaker\_job['tasks'] = tie\_tasks  
 tie\_breaker\_jobs = {'test tie breaker job': tie\_breaker\_job}  
 return tie\_breaker\_jobs  
  
  
def publish\_tie\_breaker\_job\_for\_labeling(tie\_breaker\_job\_data):  
 bucket\_name = 'tie-breaker-jobs'  
 file\_name = 'tie\_breaker\_jobs.json'  
 s3\_resource.Object(bucket\_name, file\_name).put(Body=json.dumps(tie\_breaker\_job\_data))  
  
# Function which processes output manifest and based on labeler's response it either creates tie breaking job or writes  
# final combined manifest to S3 bucket  
  
def process\_labels(output\_manifests):  
 for output\_manifest in output\_manifests:  
 tasks\_to\_be\_created = []  
 for sentence, selections in output\_manifest.items():  
 results = []  
 for selection in selections:  
 for result in selection.values():  
 results.append(result)  
  
 # Check whether labelers response is same or not  
 is\_same = all(element == results[0] for element in results)  
 if is\_same:  
 final\_labeling\_job\_result(results, selections, sentence)  
 else:  
 tasks\_to\_be\_created.append(sentence)  
  
 if len(tasks\_to\_be\_created) <= 0:  
 write\_final\_output\_tos3('combined-output-manifest', finalOutput)  
 else:  
 tie\_breaker\_job = create\_tie\_breaker\_job(tasks\_to\_be\_created)  
 publish\_tie\_breaker\_job\_for\_labeling(tie\_breaker\_job)  
  
# Function to get s3 bucket objects  
def list\_bucket\_objects(bucket\_name):  
 # Retrieve the list of bucket objects  
 try:  
 response = s3\_client.list\_objects\_v2(Bucket=bucket\_name)  
 except ClientError as e:  
 logging.error(e)  
 return None  
 return response['Contents']