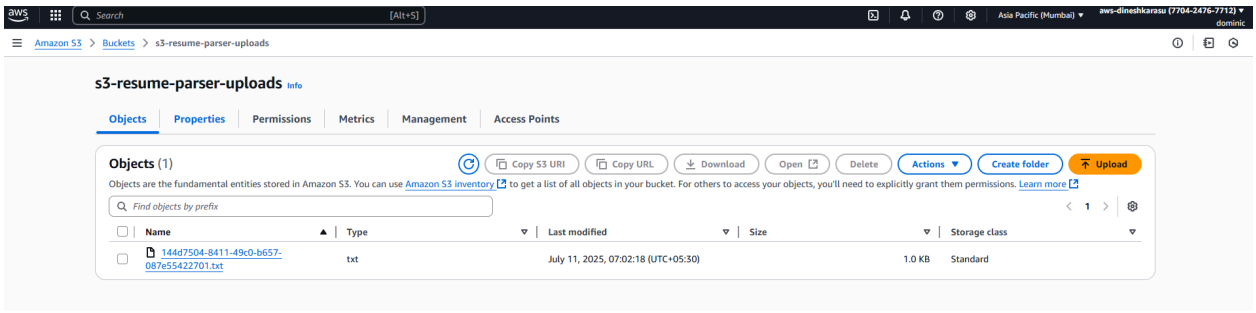
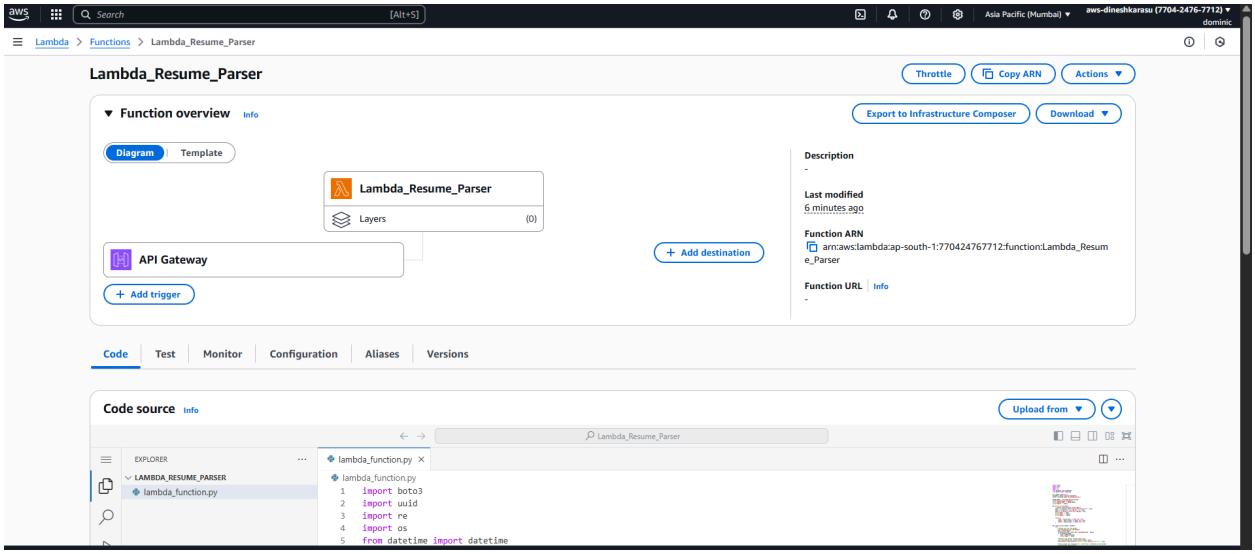
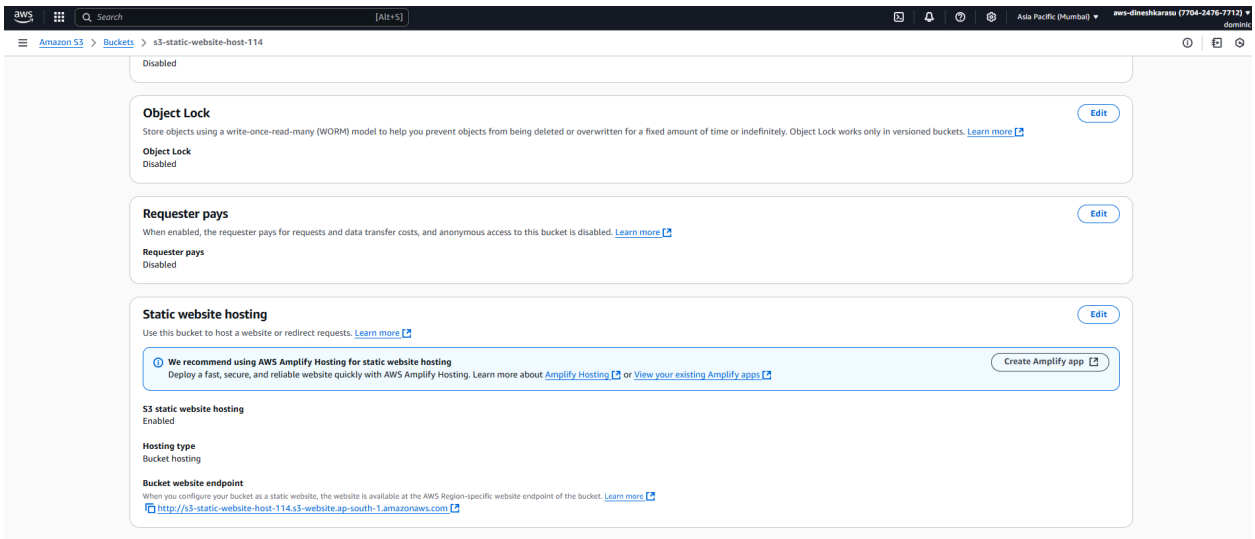
Use case	Use case Description
Mini-project: Resume Parser	Upload resume file and parse it then store it in db

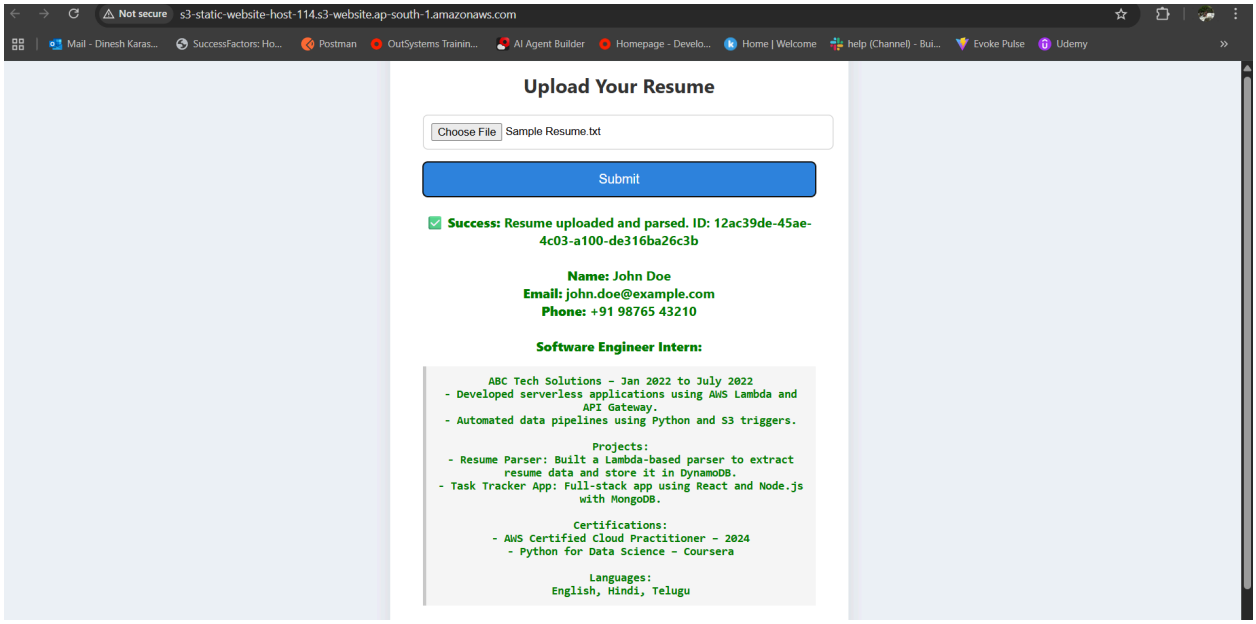
Approach :

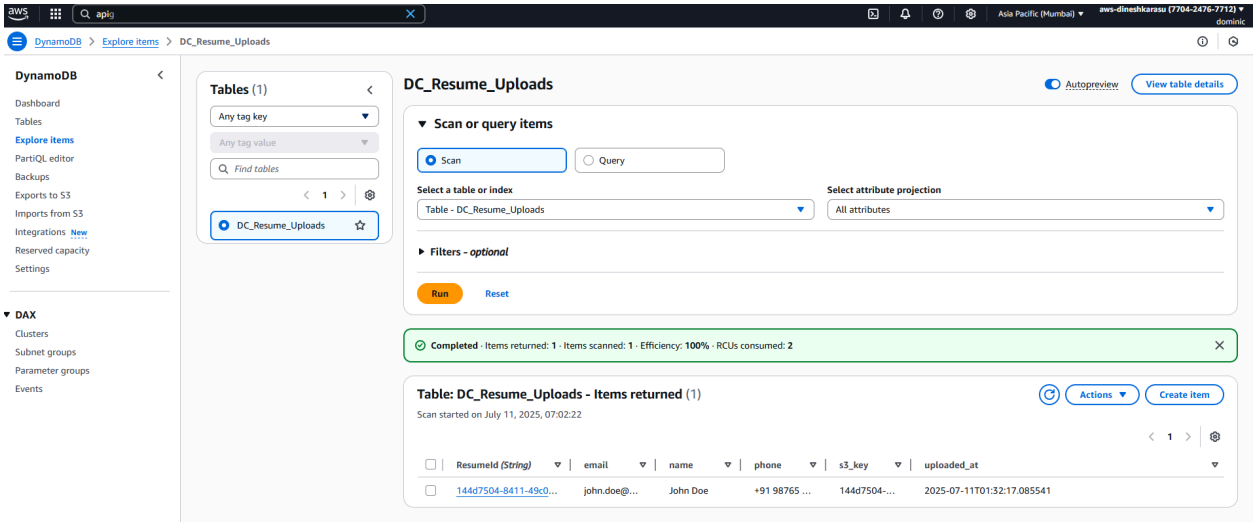
1. Create S3 bucket to store uploaded resume files
2. Create a dynamo db table to store parsed data
3. Create a Lambda function to process the uploaded file and parse it then store it in db
4. Create API Gateway POST method with CORS enabled
5. Code front end and JS to hit the API from the end user screen
6. Provide Lambda with proper IAM permissions
7. Test the flow

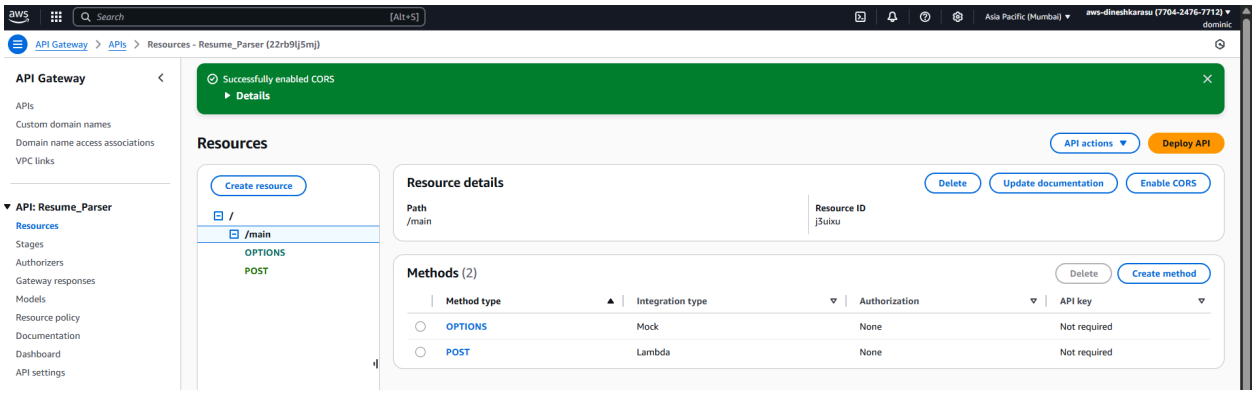


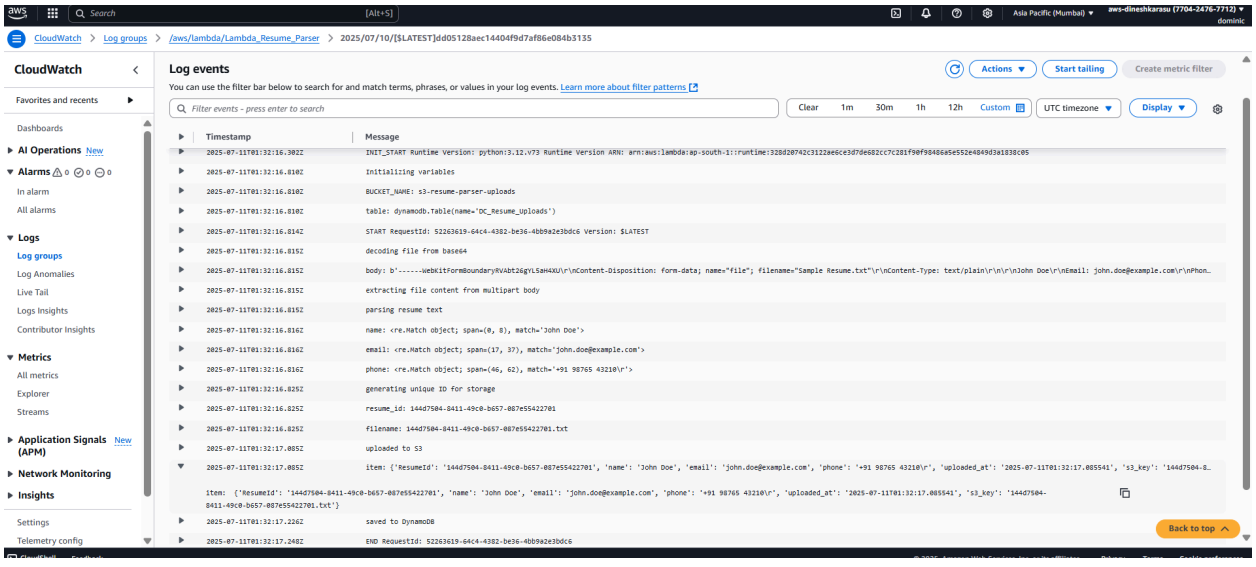

Search
[Alt+S]
Asia Pacific (Mumbai)
aws-dineshkarasu (7704-2476-7712)
dominic
Lambda > Functions > Lambda_Resume_Parser
Lambda_Resume_Parser
Throttle
Copy ARN
Actions
Function overview Info
Export to Infrastructure Composer
Download
Diagram
Template
Lambda_Resume_Parser
Layers (0)
API Gateway
Add destination
Add trigger
Description
-
Last modified
6 minutes ago
Function ARN
arn:aws:lambda:ap-south-1:770424767712:function:Lambda_Resume_Parser
Function URL Info
-
Code
Test
Monitor
Configuration
Aliases
Versions
Code source Info
Upload from
EXPLORER
LAMBDA_RESUME_PARSER
lambda_function.py
import boto3
import uuid
import re
import os
from datetime import datetime


Search
[Alt+S]
Asia Pacific (Mumbai)
aws-dineshkarasu (7704-2476-7712)
dominic
Amazon S3 > Buckets > s3-static-website-host-114
Disabled
Object Lock
Edit
Store objects using a write-once-read-many (WORM) model to help you prevent objects from being deleted or overwritten for a fixed amount of time or indefinitely. Object Lock works only in versioned buckets. Learn more
Object Lock
Disabled
Requester pays
Edit
When enabled, the requester pays for requests and data transfer costs, and anonymous access to this bucket is disabled. Learn more
Requester pays
Disabled
Static website hosting
Edit
Use this bucket to host a website or redirect requests. Learn more
We recommend using AWS Amplify Hosting for static website hosting
Deploy a fast, secure, and reliable website quickly with AWS Amplify Hosting. Learn more about Amplify Hosting or View your existing Amplify apps
Create Amplify app
S3 static website hosting
Enabled
Hosting type
Bucket hosting
Bucket website endpoint
When you configure your bucket as a static website, the website is available at the AWS Region-specific website endpoint of the bucket. Learn more
http://s3-static-website-host-114.s3-website.ap-south-1.amazonaws.com

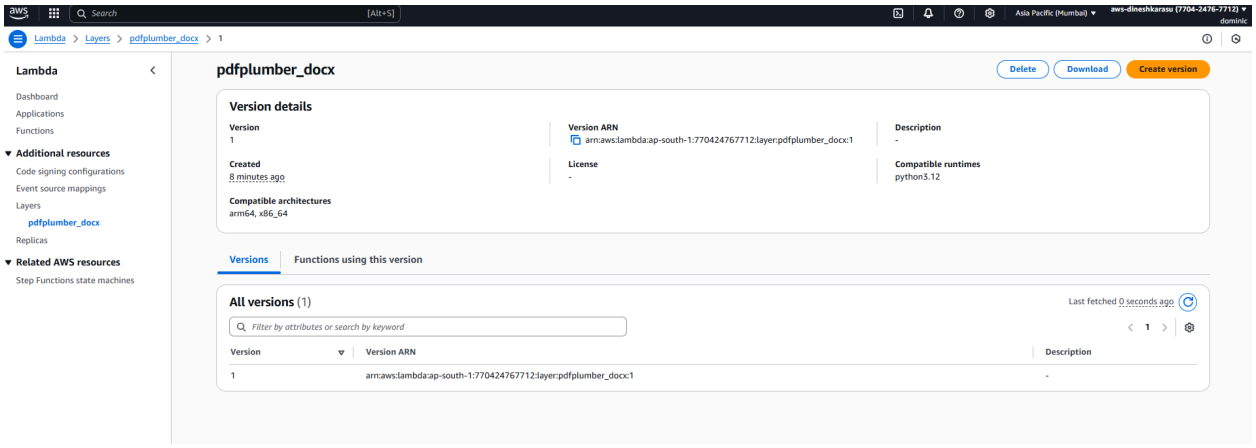

Not secure
s3-static-website-host-114.s3-website.ap-south-1.amazonaws.com
Mail - Dinesh Karas...
SuccessFactors: Ho...
Postman
OutSystems Trainin...
AI Agent Builder
Homepage - Develo...
Home | Welcome
help (Channel) - Bui...
Evoke Pulse
Udemy
Upload Your Resume
Choose File
Sample Resume.txt
Submit
Success: Resume uploaded and parsed. ID: 12ac39de-45ae-4c03-a100-de316ba26c3b
Name: John Doe
Email: john.doe@example.com
Phone: +91 98765 43210
Software Engineer Intern:
ABC Tech Solutions – Jan 2022 to July 2022
- Developed serverless applications using AWS Lambda and API Gateway.
- Automated data pipelines using Python and S3 triggers.
Projects:
- Resume Parser: Built a Lambda-based parser to extract resume data and store it in DynamoDB.
- Task Tracker App: Full-stack app using React and Node.js with MongoDB.
Certifications:
- AWS Certified Cloud Practitioner – 2024
- Python for Data Science – Coursera
Languages:
English, Hindi, Telugu


DynamoDB
Explore items
DC_Resume_Uploads
Tables (1)
DC_Resume_Uploads
Autopreview
View table details
Scan or query items
Scan
Query
Select a table or index
Table - DC_Resume_Uploads
Select attribute projection
All attributes
Filters - optional
Run
Reset
Completed · Items returned: 1 · Items scanned: 1 · Efficiency: 100% · RCUs consumed: 2
Table: DC_Resume_Uploads - Items returned (1)
Scan started on July 11, 2025, 07:02:22
Actions
Create item
ResumeId (String)
email
name
phone
s3_key
uploaded_at
144d7504-8411-49c0...
john.doe@...
John Doe
+91 98765 ...
144d7504-...
2025-07-11T01:32:17.085541


API Gateway
APIs
Resources - Resume_Parser (22rb9lj5mj)
Successfully enabled CORS
Details
Resources
API actions
Deploy API
Create resource
/
/main
OPTIONS
POST
Resource details
Delete
Update documentation
Enable CORS
Path
/main
Resource ID
j3uixu
Methods (2)
Delete
Create method
Method type
Integration type
Authorization
API key
OPTIONS
Mock
None
Not required
POST
Lambda
None
Not required


CloudWatch
Log groups
/aws/lambda/Lambda_Resume_Parser
2025/07/10/[$LATEST]dd05128aec14404f9d7af86e084b3135
Log events
Actions
Start tailing
Create metric filter
You can use the filter bar below to search for and match terms, phrases, or values in your log events.
Learn more about filter patterns
Filter events - press enter to search
Clear
1m
30m
1h
12h
Custom
UTC timezone
Display
Timestamp
Message
INIT_START Runtime Version: python:3.12.v73 Runtime Version ARN: arn:aws:lambda:ap-south-1::runtime:328d20742c3122ae6ce3d7de682cc7c281f90f98486a5e552e4849d3a1838c05
Initializing variables
BUCKET_NAME: s3-resume-parser-uploads
table: dynamodb.Table(name='DC_Resume_Uploads')
START RequestId: 52263619-64c4-4382-be36-4bb9a2e3bdc6 Version: $LATEST
decoding file from base64
extracting file content from multipart body
parsing resume text
name: <re.Match object; span=(0, 8), match='John Doe'>
email: <re.Match object; span=(17, 37), match='john.doe@example.com'>
phone: <re.Match object; span=(46, 62), match='+91 98765 43210\r'>
generating unique ID for storage
resume_id: 144d7504-8411-49c0-b657-087e55422701
filename: 144d7504-8411-49c0-b657-087e55422701.txt
uploaded to S3
saved to DynamoDB
END RequestId: 52263619-64c4-4382-be36-4bb9a2e3bdc6
Back to top

Since the above approach only works for .txt files but not pdf and docx files as we are using regular expressions, let's go with another approach using python-docx
We'll keep everything as it is and modify our lambda code. Create a layer then update the lambda code



lambda_function.py

```python
def extract_fields(text):
    email = re.search(r'[\w\.-]+@[\w\.-]+', text)
    phone = re.search(r'\+?\d[\d\s\-]{9,14}', text)

    return {
        'name': name.group() if name else 'N/A',
        'email': email.group() if email else 'N/A',
        'phone': phone.group() if phone else 'N/A',
        'sections': sections
    }

def extract_text_from_pdf(file_path):
    with pdfplumber.open(file_path) as pdf:
        return "\n".join([page.extract_text() or "" for page in pdf.pages])

def extract_text_from_docx(file_path):
    doc = docx.Document(file_path)
    return "\n".join([para.text for para in doc.paragraphs])

def lambda_handler(event, context):
    try:
        body = event['body']
        is_base64_encoded = event.get('isBase64Encoded', False)
        if is_base64_encoded:
            body = b64decode(body)

        # Extract file content from multipart body
        file_binary = body.split(b'\r\n\r\n', 1)[1].rsplit(b'\r\n------', 1)[0]

        # Extract filename from headers
        header_line = body.split(b'\r\n', 2)[1]
        filename_match = re.search(b'filename="(.+?)"', header_line)
        filename = filename_match.group(1).decode() if filename_match else f"{uuid.uuid4()}.txt"
        ext = filename.lower().split('.')[-1]

        # Save to temp file for parsing
        with tempfile.NamedTemporaryFile(delete=False, suffix=f".{ext}") as tmp:
            tmp.write(file_binary)
            tmp_path = tmp.name

        # Parse file content
        if ext == 'pdf':
            text = extract_text_from_pdf(tmp_path)
```

Ln 29, Col 23 Spaces: 4 UTF-8 LF Python Lambda Layout: US

lambda_function.py

```python
def lambda_handler(event, context):
    try:

        item = {
            'ResumeId': resume_id,
            'name': parsed['name'],
            'email': parsed['email'],
            'phone': parsed['phone'],
            'uploaded_at': datetime.utcnow().isoformat(),
            's3_key': s3_key,
            'sections': parsed['sections']
        }
        table.put_item(Item=item)

        cleaned_sections = {}
        main_fields = {parsed['name'].lower(), parsed['email'].lower(), parsed['phone'].replace(" ", "")}
        for section_title, section_text in parsed['sections'].items():
            lower_text = section_text.lower().replace(" ", "")
            if not any(field in lower_text for field in main_fields):
                cleaned_sections[section_title] = section_text
        parsed['sections'] = cleaned_sections

        return {
            'statusCode': 200,
            'headers': {
                'Access-Control-Allow-Origin': '*',
                'Access-Control-Allow-Methods': 'POST, OPTIONS',
                'Access-Control-Allow-Headers': 'Content-Type'
            },
            'body': json.dumps({
                'message': f"Resume uploaded and parsed. ID: {resume_id}",
                'parsed': {
                    'name': parsed['name'],
                    'email': parsed['email'],
                    'phone': parsed['phone'],
                    'sections': parsed['sections']
                }
            })
        }

    except Exception as e:
        traceback.print_exc()
        return {
```

Ln 29, Col 23 Spaces: 4 UTF-8 LF Python Lambda Layout: US