**Technical Interview**

**Python Script**

personal\_information = [

{"last\_name": "Smith", "full\_name": "John Smith", "age": 30, "location": "California"},

{"last\_name": "Williams", "full\_name": "Rebecca Williams", "age": 25, "location": "Texas"},

{"last\_name": "Miller", "full\_name": "Juanita Martina Mary Miller", "age": 35, "location": "Toronto"},

{"last\_name": "Brown", "full\_name": "Paul A. Brown", "age": 35, "location": "England"},

{"last\_name": "Brown", "full\_name": "Paula Williams", "age": 20, "location": "England"},

]



Implement a function that returns a dictionary where the keys are locations, and the values are lists of full names of people from that location and sum of their ages. For example, the output should be:

{

"California": (["John Smith"], 30),

"Texas": (["Rebecca Williams"], 25),

"Toronto": (["Juanita Martina Mary Miller"], 35),

"England": (["Paul A. Brown", “Paula Williams”], 55)

}

dictionary = {}

for info in personal\_information:

# {"last\_name": "Smith", "full\_name": "John Smith", "age": 30, "location": "California"}

key = info['location']

full\_name = info['full\_name']

age = info['age']

if key in dictionary:

(names, sum\_of\_ages) = dictiory[key]

names.append(full\_name)

new\_names = names

sum\_of\_ages += age

dictionary[key] = (new\_names, sum\_of\_ages)

else:

dictionary[key] = ([full\_name],age)



Intelligent Document Processing

You need to design and build a **scalable, modular, and extensible AI-driven document processing pipeline**. This pipeline should be capable of

1. Ingesting various document types - Laws, Invoices, OCR, Large PDFs
2. **Generative AI tasks** such as Summarization, Named Entity Recognition (NER), Question Answering (QA), Sentiment Analysis - ML Developers
3. Seamlessly **Integrate multiple products**
4. Efficient processing, and robust error handling

Three major components -

1. Data Processing
2. ML / Gen AI Server
3. Post Processing

API

Fast API ->

Scala Play Framework

POST files -> POST async /upload?

GET status Instant response -> id=1

GET summarised\_results -> id -> get id and completed -> fetch data -> return

1. -> Different Response Times [Less Way too 5 minutes -> Event driven]
2. -> Different Accesses

Microwaveservices ()

Ingested

Processing

Completed

Apache Kafka

-> {

}

NiFi Flow (Ingested)

* >

Database

Ingested

Processing

Completed

-> ElasticSearch {

}

AWS Services

S3 -> ETag ->