|  |
| --- |
| Intelligent Invoice Processing System  Solution Design Approach |

**Step-1**

In this step it shows how the business problem is solved with utmost efficiency by using various techniques to reach the goal. Here Textract is being used which is an AWS service though which we can perform text extraction.

1. The invoices will be uploaded through the front-end through an API call and that will store the documents in a specified S3 bucket
2. The uploading can be a single document or batches of documents.
3. After the documents have been uploaded to S3 bucket a Lambda function will be triggered
4. This Lambda function is for Pre-Processing the document where the document will be refined and then stored to another S3 bucket.
5. After the pre-processed document Is stored in the S3 bucket another Lambda function will be triggered.
6. This triggered lambda function is for Post-Processing of the document.
7. Here the document is processed by calling Textract API functions on the document where the it will be sent to textract for processing and will be getting a json response back from textract.
8. There can be 2 outcomes-
   1. The successful invoices
   2. Flagged invoices
   3. The successful invoices will be stored in S3 as a JSON file.
   4. The flagged invoices will go to other S3 bucket folder.
9. After the json output is uploaded In S3 bucket it then will trigger another lambda function which will take this json output and save it in DynamoDB.
10. After saving the data in DynamoDB following information will be displayed on the screen-
    1. Header information: Supplier Name, Invoice No., Invoice Date, GSTN no.
    2. Footer Information: Taxes & Invoice Amount
    3. Line level: Item level information like part no., qty & amount
11. After this we will have to send data from DynamoDB to an ERP and for that we have few options-
    1. API option
    2. DMS
    3. Python script from SQL
    4. CSV import and export
12. Here we chose API option so whenever anyone hits the API a lambda function will create a DB request which will pull the data from DynamoDB and provide it to the requested person.

**Step-2- Augmented AI functionality**

After post-processing of invoice, we can get 2 outcomes-

* + The successful invoices/ data properly extracted
  + Flagged invoices/ not properly extracted

Flagged invoices will be rectified using A2I which is a service built by AWS.

A2I starts as a human loop which works as a correction mechanism for flagged invoices.

1. The output that will be given by A2I will be saved into a S3 bucket which then will trigger another lambda function.
2. This lambda function will parse A2I output to readable json format as it contains nested json with other unwanted data
3. The data will be stored in the same S3 bucket where the success invoice outputs are stored for further process

**Step-3-enabling different invoice templates**

Creating a solution for different kinds of invoice templates.

Will analyze meaningful headers and contents to be extracted from the invoices.

Will change/manipulate the headers being used in old code.

**Challenges Faced and resolved**

Following are the major issues that we faced during the implementation of solution :

1. Pillow library issue – For preprocessing the PDF in lambda we used a function present in the Pillow(PIL) library. We used a lambda layer to import this library in our lambda. Still the functions of the library were not working. After some research we figured out the statement to import the library had to be written in a specific format.

2. pdf2image and image\_cleaner module issue - The library was not available in lambda. Thus, we had to create an EC2 Linux instance and install the library in a python virtual environment. Then we copied the library files from the instance and saved them in a lambda layer. This allowed us to import pdf2image.

3. Poppler library issue – we tried to use the functionality of pdf2image library, but it required us to install poppler in our system , as the former is just a wrapper around the latter. That means to use pdf2image we need to have poppler installed in the backend.

4. img2pdf attribute issue – this issue was of not being able to import the library but was resolved by modifying the import call.

**Test Cases**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case No.** | **Description** | **Test Data** | **Expected Result** | **Actual Result** | **Status (pass/fail)** |
| #1 | When an invoice with multiple marks and cuts on the image is uploaded | Invoice with multiple cuts and other marks | The invoice data must be saved in Failed invoices s3 bucket | The data on the invoice is saved in the failed invoice s3 bucket | Pass |
| #2 | A clean invoice having multiple line items gets all its data saved in DynamoDB. | An invoice with a table that contains multiple line items data | All the records of the table present in the invoice are saved in DynamoDB table | The data of the invoice table is saved in DynamoDB and can be viewed in ERP | Pass |

Solution Summary - Training & Handover Notes

**About Challenge(s)**

The business problem was that, MATE’s back-office operations team, process Invoices as a part of back-office operations for their customers, all these invoices which they receive from their customers are manually processed.

In this manual processing they pick information manually from scanned invoice documents and put in system, below information is extracted from invoices –

* Header information (e.g. Supplier Name, Invoice No., Invoice Date, GSTN No etc.)
* Footer information (e.g. Taxes & Invoice Amount etc.)
* Line level information: Item level information (e.g. part no., Qty, amount etc.)

This is quite time consuming, error prone and tedious for them to accomplish for nearly 30K Invoices every month, a team of 12 people are engaged in this processing leads to lot undifferentiated activity which creates scope of automation. Other OCR solutions which were being tried had their own shortcomings so couldn’t be adopted as a solution.

**Proposed Solution**

It is required to have an automated serverless solution to address this requirement. So, all the invoices are copied to a pre-decided system location and our system picks from that location and extract the required information from Invoice documents using Amazon Textract, as  it is more than a simple optical character recognition (OCR) to also identify the contents of fields in forms and information stored in tables, and put into database system and process all invoices one by one. Steps involved are as follows -

* Pre-processing of Invoices – to clean invoice documents which increases readability
* Post-processing of Invoices – to extract information and validate data to further improve quality of solution and made it error-proof also, line/word confidence level to make it complete solution
* Creating a human loop – using Amazon A2I to review and edit flagged invoices
* Updating data to ERP – sending DynamoDB data to SQL server where all the information is priorly stored]

**AWS Services used**

* Amazon Textract – Extract Data from Scanned Documents
* API Gateway – Rest API which calls a Lambda function
* Lambda – Integrated code to call multiple API’s
* S3 – Storing the raw, pre & post processed files

**Solution Outcome**

All in all, we successfully created a solution to process invoices with accuracy of 99% approx. This resulted in manual effort saving of about 40% of team (FTE’s) initially and thus lead to

* Reduction in effort and cost. (40% team reduced)
* Reduction in cycle time
* Improvement in quality of processing

There is increased customer satisfaction which possibly will be translated to more orders from same customer.

**Architecture Diagram**

Diagram

Description automatically generated

**How AWS services helped in building the model for sales Forecasting**

**Amazon S3 to store files in different stages**

It is an object storage service that offers industry-leading scalability, data availability, security, and performance. In this solution it helped to store raw invoice documents, preprocessed invoice documents and for storing final JSON output files.

**AWS Lambda to run code serverless for preprocessing and post processing**

It let us run code without provisioning or managing servers. Lambda allowed us to run code for virtually any type of application or backend service - all with zero administration. Just upload the code and Lambda takes care of everything required to run and scale the code with high availability. Here, preprocessing lambda and post processing lambdas were set to be automatically triggered from different S3 buckets.

**DynamoDB to store key-value pairs from processed JSON**  
It is a key-value and document database that delivers single-digit millisecond performance at any scale. It's a fully managed, multiregional, multi-master, durable database with built-in security, backup and restore, and in-memory caching for internet-scale applications. It helped to store processed JSON for invoice documents.

**Amazon Textract to extract data from PDF**

It is a fully managed machine learning service that automatically extracts printed text, handwriting, and other data from scanned documents that goes beyond simple optical character recognition (OCR) to identify, understand, and extract data from forms and tables. Here, it helped to extract data from scanned invoice documents format. It uses machine learning to instantly read and process any type of document, accurately extracting printed text, handwriting, forms, tables and, other data which helped in quickly automating manual document related activities.