Bulk Document Parsing

Overview:

The project involved parsing a PDF to extract required data and store it in the database for further use. The client is a clinical research organization and supports the medical industry with data research and regulatory submissions. The solution enabled the client to extract the required information: key-value pairs and tabular data and store it in the repository.

Client details:

Name: Confidential | **Industry:** Transportation & Logistics | **Location:** USA

Technologies:

Pytesseract, opencv, invoice2data, azure-storage, celery, RabbitMQ

Modules Used:

- **Pytesseract:** This is used in Non-Searchable PDF parsing. A python wrapper around Google's Tesseract performs Optical Character Recognition; i.e. gives a string output for text in images.
- **OpenCV**: This is used in Non-Searchable PDF parsing. It performs Computer Vision operations on an image.
- **invoice2data:** This is used in Searchable PDF parsing. It uses YAML templates to parse the data from searchable PDFs and returns the result as CSV, JSON or XML.
- **Azure-storage**: It is used for storing the PDFs as we needed a structured storage. If the requirement is an unstructured storage then Azure Blob Storage was used.
- **Celery & RabbitMQ:** It is used to handle processing of tens of thousands of PDFs using workers and message queue. Multiple queues with different priorities help to process high priority PDFs as soon as possible.

Project Description:

Earlier, the client used to manually search through previous PDF Documents to filter out its requirements. It lead to cost overruns and was inefficient. After analyzing the client's need, team@Mindfire offered to develop a solution that would help the client to parse a PDF and extract the relevant info. There are two modules available to parse PDFs. Some salient features are:

Searchable PDF:

- Searchable PDFs result from the application of OCR (Optical Character Recognition) to scanned PDFs or other image-based documents.
- The solution parses invoices of different companies using the YAML Template. Python's invoice2data library uses YAML templates (containing keys and the respective regex to map the values to these keys) to extract key-value pairs and uses the JSON data for further operations.

Non- Searchable PDF:

- Non- Searchable PDFs are documents that cannot be searched by text and are created by saving the PDF file as an image.
- HTML 5 Canvas marks the dimensions of the data to extract, and the JSON File stores the individual data dimension.
- For parsing, the PDF is converted to an image. OpenCV reads the image and contour detection to identify the text.
- JSON Template locates the data where images appear, and the OpenCV uses filter operations to operate on the cropped image.
- Tesseract processes the cropped image for string output which, in turn, is stored in a JSON file for further operations.

After the initial launch of the application, the team faced two major challenges:

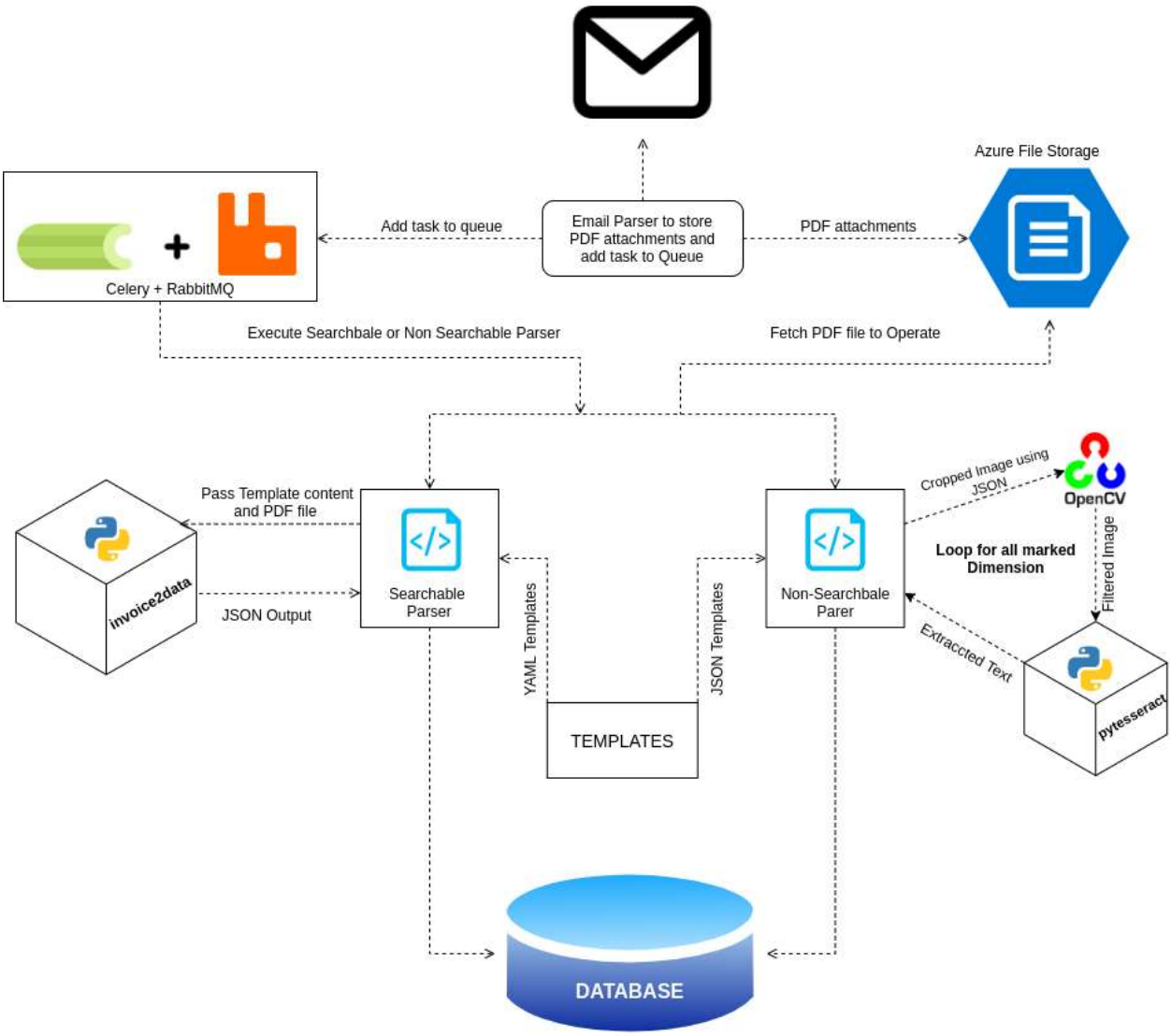
OCR Output Efficiency

- Earlier, the accuracy of the output was 80% as the OCR faced difficulties in interpreting certain letters like B, 3, S, $, O, 0 etc.
- The accuracy was improved by the use of image processing before sending the image to OCR Engine. This resulted in excellent results with 99% accuracy.

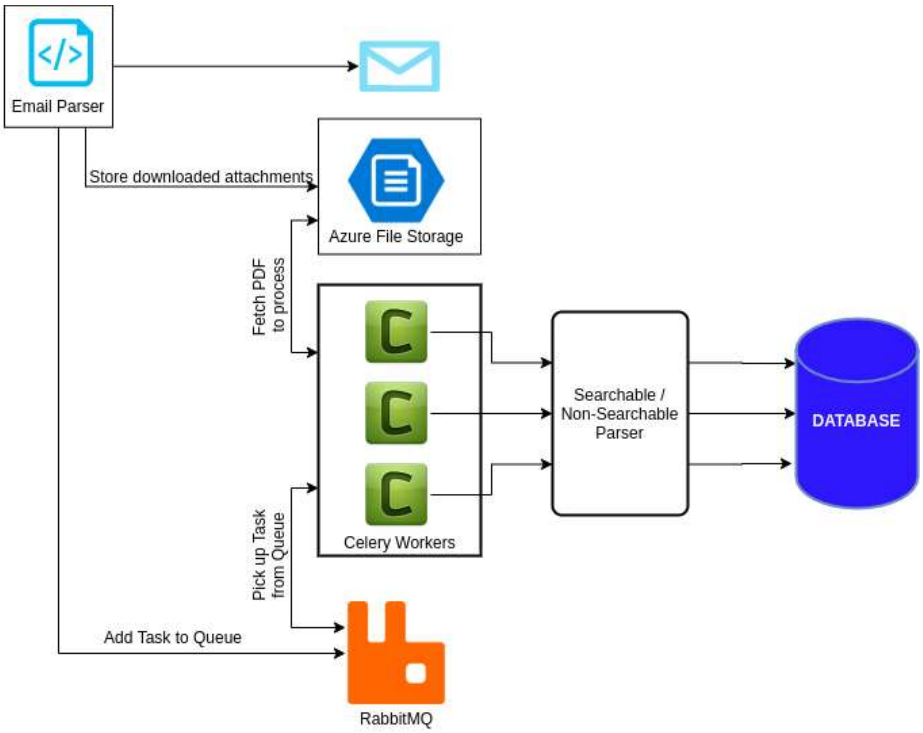
Scalability

- After the initial launch of the application, the time taken to parse a single PDF page was approximately 3 sec. The client's requirement was about 10,000 pages a day, and the current speed did not suffice.
- The processing speed was improved with the help of multiprocessing. The team implemented this with the help of RabbiMQ, a queue server.
- The RabbiMQ parses the invoices in parallel. The application picks up the attachments from the mail, stores them in the azure server, and adds the task in the queue. The module picks up the pending tasks from the queue and processes them in parallel.

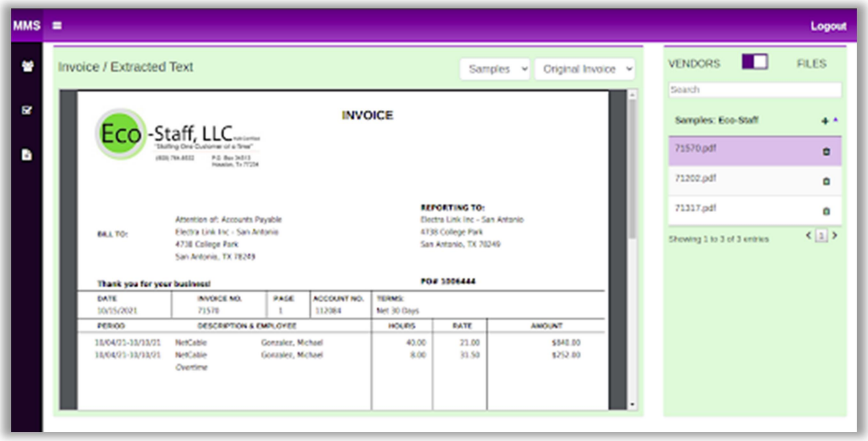
Architecture:



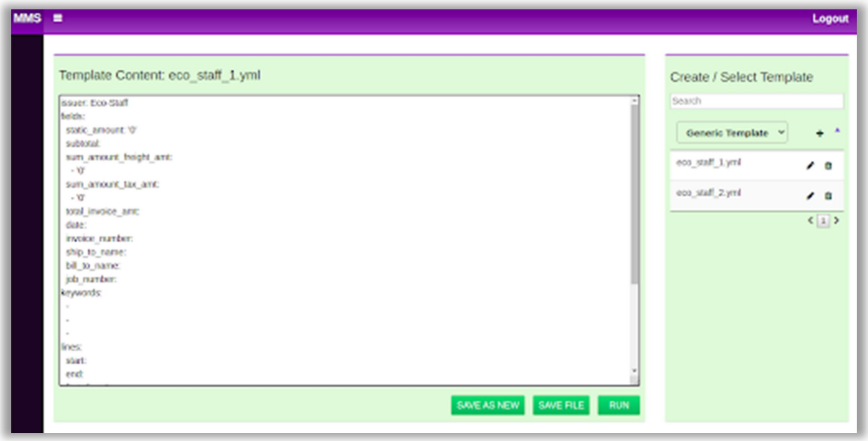
Workflow:



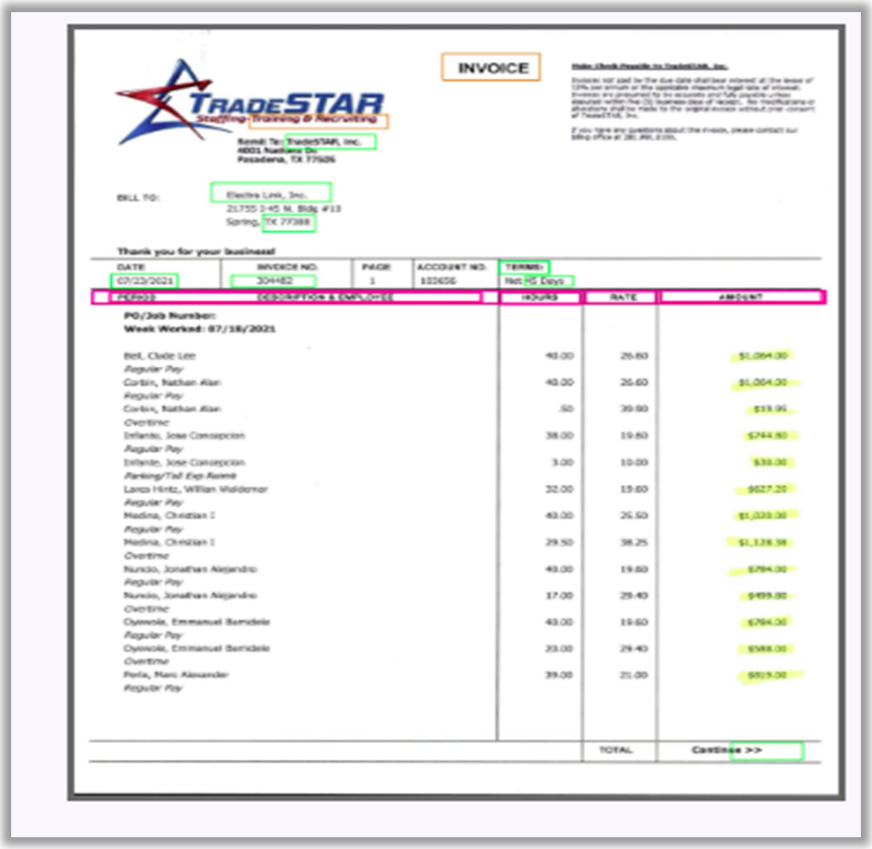
Screenshots:



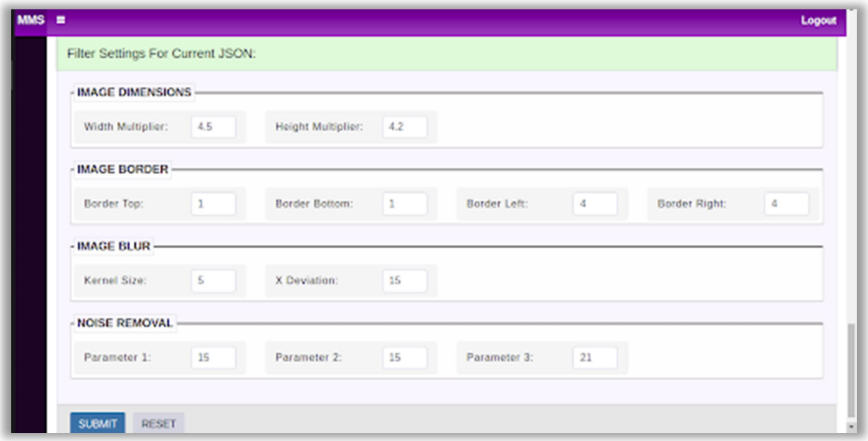
Screenshot 1: YAML is created for a particular vendor based on the format



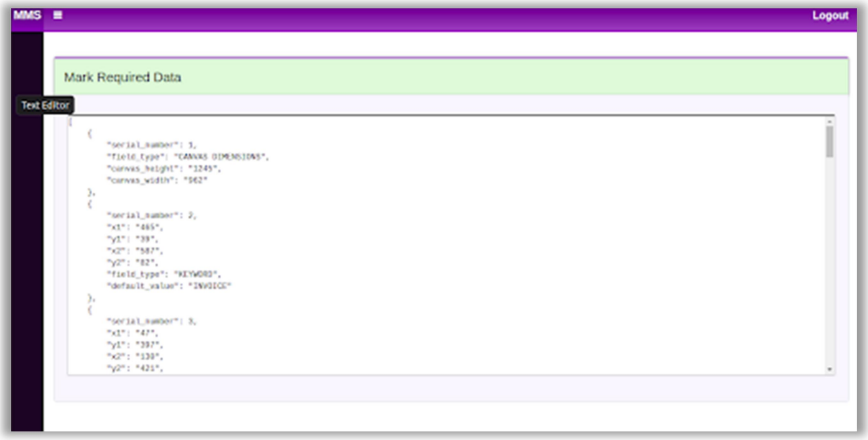
Screenshot 2: Searchable Parser: Template. It has the required fields created by users in YAML according to invoice format.



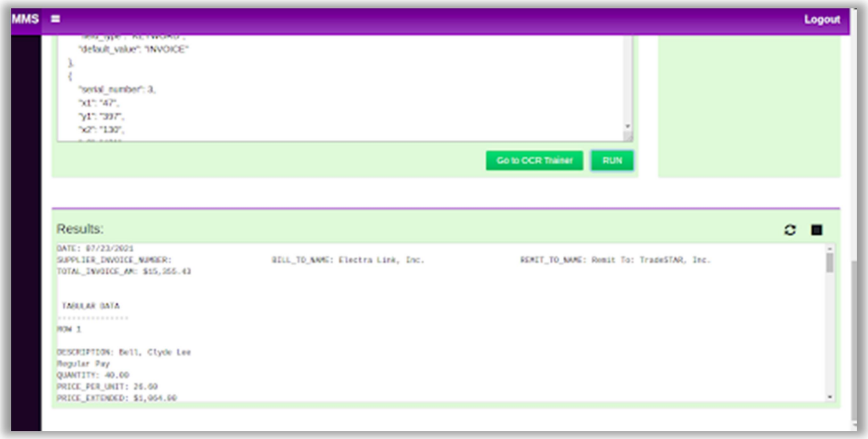
Screenshot 3: Marked invoice



Screenshot 4: Filter Settings for JSON



Screenshot 5: Marked Dimensions: JSON Template



Screenshot6: Processed data