**Production OCR Pipeline –**

Two Pipeline:

1. General IRIS Pipeline
2. Vendor Onboarding Pipeline

**General IRIS Pipeline:**

**Files used (in order of execution):**

1. text\_extraction.py and raw\_text\_extraction.py
2. ocr\_pretrained.py
3. ocr\_custom.py
4. merge.py
5. callback.py

**text\_extraction.py: (vwits/iris/cv/services/text\_extraction\_service/text\_extraction.py)**

* The input file is a pdf/image.
* If input file is pdf then each page of pdf is converted to image(max 3 pages of pdf) first.
* If the converted image is large than 4mb, then this image is first compress to fit to azure acceptable dimensions size.
* This file majorly focus on extracting the text out of this image files and return a json object with bounding box details and raw text output.

**raw\_text\_extraction.py:  (vwits/iris/cv/services/invoice\_service/raw\_text\_extraction.py)**

* This file takes the output from above file and insert bounding box details to ocr\_result collection and raw text output to raw\_text\_output collection.
* On successful insertion, update process\_id in files collection to 2 else -1.

Collections used: files, ocr\_result, raw\_text\_output.

**ocr\_pretrained.py: (vwits/iris/cv/services/invoice\_service/ocr\_pretrained.py)**

* This file contains the code for running the inhouse OCR pretrained model.
* The input file is a pdf/image and return a json object representing key information

such as vendor name, po number, amount before tax, amount after tax, etc.

* Some validation has been applied to fields like PO number, amount before tax, amount after tax.

**ocr\_custom.py: (vwits/iris/cv/services/invoice\_service/ocr\_custom.py)**

* This file contains the code for running the inhouse OCR Custom model.
* The input file is a pdf/image and return a json object representing extracted Vendor GST number.

**merge.py: (vwits/iris/cv/services/invoice\_service/merge.py)**

* This file contains the code to merge the output of ocr\_custom model with ocr\_pretrained model.
* The final merged output is then stored in azure\_linking collection.
* On successful insertion, update process\_id in files collection to 3 else -1.

Collections used: files, azure\_linking

**callback.py: (vwits/iris/cv/services/invoice\_service/callback.py)**

* This file contains the code for sending the callback to IRIS.
* Total 4 URLS for Vwits, Savwipl, Avme, Vendor\_onboarding.
* For IRIS pipeline: Update process\_id in files collection to 4 if callback sent successfully else -1.
* For Vendor pipeline: Update process\_id in files collection to 7 if callback sent successfully else -1.

Collections used: files

**Vendor Onboarding Pipeline:**

**Files used:**

1. pdf\_to\_image.py
2. image\_preprocessing.py
3. id\_doc.py
4. vendor\_merge.py

**pdf\_to\_image.py: (vwits/iris/cv/services/id\_doc\_service/pdf\_to\_image.py)**

* This file contains the code to convert the PDF files into images (JPG).
* PDF, JPG or PNG files are accepted and handled currently for input.

**image\_preprocessing.py: (vwits/iris/cv/services/id\_doc\_service/image\_preprocessing.py)**

* This file contains the code to remove the white border from the pdf like PAN card.
* JPG or PNG files are accepted and handled currently for input.
* If PDF is given as input, then using pdf\_to\_image file above, it is converted to image first.
* This Is majorly used for PAN documents.

**id\_doc.py: (vwits/iris/cv/services/id\_doc\_service/id\_doc.py)**

* This file contains the code to extract text from identity documents like PAN,GST,MSME,Bank.
* This uses the model id for the trained model stored in .env file. (/home/Ubuntu/ashutosh/research /inhouse\_ocr/vwits/iris/cv/services/.env)
* Return json data which includes key information for each respective documents.

**vendor\_merge.py: (vwits/iris/cv/services/id\_doc\_service/vendor\_merge.py)**

* This file takes the output from above file and insert into vendorData collection.
* On successful insertion, update process\_id in files collection to 6 else -1.

Collections used: files, vendorData

**Use of Each Collection:**

1. files– Contains information related to invoice like case\_id, invoice name, process\_id.
2. azure\_linking – Contains the final merged output of ocr\_pretrained model and ocr\_custom model. Final output of product is stored in this collection.
3. entity\_linking – Contains the page wise linked entities
4. ocr\_result – Contains bounding box details extracted from each invoices.
5. raw\_text\_output – Contains raw text extracted from each invoices.
6. refreshtokens – Used for authorized token of users.
7. vendorData – Contains extracted text for each identity documents.

**4.       Folder used:**

* + files:
  + Path : /home/ubuntu/ashutosh/research/inhouse\_ocr/vwits/iris/cv/image\_processing/data/files
  + Description : Stores all the invoice images from an API call
  + image\_archive:
  + Path /home/ubuntu/ashutosh/research/inhouse\_ocr/vwits/iris/cv/image\_processing/data/image\_archive
  + Description : Backup of  all the invoice pdf/images from an API call
  + image\_dump :
* Path : /home/ubuntu/ashutosh/research/inhouse\_ocr/vwits/iris/cv/image\_processing/data/image\_dump
* Description : Backup of  all the converted images.

**In-House OCR API Integration:**

Use two files:

* **app\_concurrency.py: (vwits/flask\_module/app\_concurrency.py)**
  + This files contains all the API of OCR.
* **authorization.py : (vwits/flask\_module/authorization.py)**
  + This file is used to authorize the API of OCR using bearer token.
  + The bearer token for each user are inserted to refreshtokens collection
  + While using API of OCR, first the token is verified and its expiry date is checked. Once valid users can access the API else gives unauthorized message.

1) **Post a file for scanning :**

For IRIS Pipeline:

* URL: [http:// 65.1.240.54:5000/create](http://34.232.250.208:5000/create)
* Method: POST
* Request:
  + file : file stream
* Response:   case id

For Vendor Pipeline:

* URL: [http:// 65.1.240.54:5000/create](http://34.232.250.208:5000/create)/<type>
* Method: POST
* Request:
  + file : file stream
  + type: type of documents (e.g. PAN, GST, Bank, MSME)
* Response: case id

2) **Get case details by process id & case id :**

* URL: [http:// 65.1.240.54:5000/get\_data/<process\_id>/<case\_id](http://34.232.250.208:5000/get_data/%3cprocess_id%3e/%3ccase_id)>
* Method: GET
* Content-Type: application/json
* Request Body : N/A

(e.g: http://65.1.240.54:5000/get\_data/3/eb4c8d58-b6ec-4864-ba72-fe65f5a04ee3)

Note: process\_id is always 3.

3) **Get case raw text by filename :**

* URL : http:// 65.1.240.54:5000/get\_data\_by\_file\_name/<case\_id>/<filename>
* Method: GET
* Content-Type: application/json
* Request Body : N/A

(e.g: [http://65.1.240.54:5000/get\_data\_by\_file\_name/ eb4c8d58-b6ec-4864-ba72 fe65f5a04ee3/1624433502796\_8000098002\_5396311025.pdf](http://65.1.240.54:5000/get_data_by_file_name/%20eb4c8d58-b6ec-4864-ba72%20fe65f5a04ee3/1624433502796_8000098002_5396311025.pdf%20)  )

4) **Get case vendor data :**

* URL : http:// 65.1.240.54:5000/get\_vendor\_data /<case\_id>
* Method: GET
* Content-Type: application/json
* Request Body : N/A

(e.g: http://65.1.240.54:5000/get\_vendor\_data/ eb4c8d58-b6ec-4864-ba72 fe65f5a04ee3 )

**Service Files used:**

Path: /etc/systemd/system

* flask\_webserver.service (Flask-Api)
* mongod.service (MongoDB Service file)

Commands to run to scheduling (system files) :

* sudo systemctl daemon-reload
* sudo systemctl enable <filename>
* sudo systemctl start <filename>
* sudo systemctl stop <filename>
* sudo systemctl restart <filename>
* sudo systemctl status <filename>

**Ports :**

**5000 :** Flask API