GridScanner - Invoice Processing

How to Run(Traditional Method):

Run all following three modules to run the Application

- GridScanner:
  - Run `pip install -r requrements.txt` to install required python libraries.
  - Export Environment Variables using `source .env`
  - Run `uvicorn app:app` to start the server.
  - Go to http://127.0.0.1:8000 to access the UI.

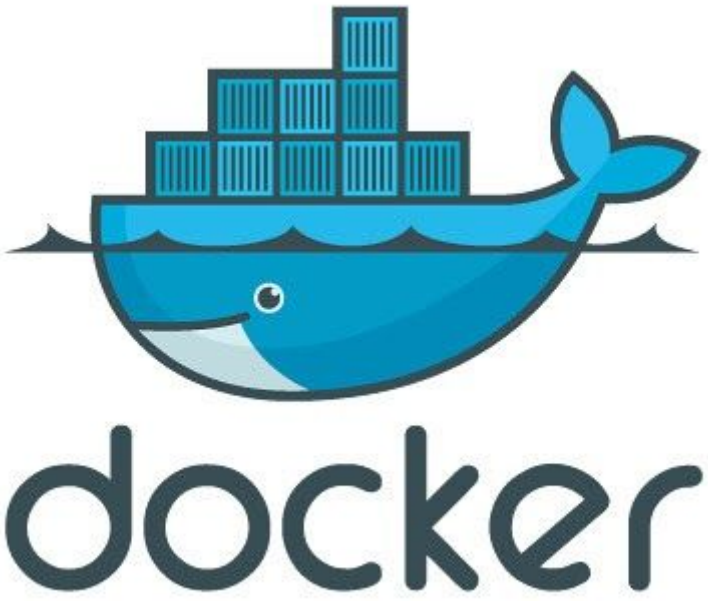
- InvoicePlaceholder:
  - `pip install -r requirements.txt`
  - Start server using `uvicorn app:app --port=8001`
  - The api is exposed at http://localhost:8001/getXLSX

- CUTIEPI:
  -
  - Run `pip install -r requrements.txt` to install required python libraries.
  - Export Environment Variables using `source .env`
  - Copy model_for_serving from Drive link. And paste into
  - main folder.
  - Start the model server using `tensorflow_model_server --port=8500 --rest_api_port=8501 --model_name=CUTIE --model_base_path="${MODEL_DIR}"`

Now go to http://127.0.0.1:8000 to access to GridScanner UI.

P.T.O for Docker Compose

Run using docker compose (Smart Method)



All three modules are packages in docker image and push to Dockerhub. So we can run using a single docker-compose file in minutes.

Steps:

1. Install Docker and Docker Compose using their official docs.
2. Create a file named docker-compose.yaml and put following contents in that

```yaml
version: "2.0"
services:
 gridscanner:
   image: jvenom/gridscanner
   environment:
     - DICT_PATH=predict/dict/grid

     - TENSORFLOW_HOST=cutiepi
     - TENSORFLOW_PORT=8500
     - TENSORFLOW_MODEL=CUTIE
     - TENSORFLOW_SIGNATURE_NAME=serving_default

     - INVOICEHOLDER_HOST=invoiceplaceholder
     - INVOICEHOLDER_PORT=8001
   ports:
     - "8000:8000"
```

```
  depends_on:
    - invoiceplaceholder
    - cutiepi
  volumes:
    - ./results:/code/results
invoiceplaceholder:
  image: jvenom/invoiceplaceholder
  ports:
    - "8001:8001"
cutiepi:
  image: jvenom/cutiepi
  ports:
    - "8500:8500"
```

3. Create results folder is same directory
4. Now run `docker-compose up` in the same directory.
5. The app will be app and running at http://127.0.0.1:8000

Tech Stack

- Tensorflow: We have use tensorflow for the Machine Learning module
- FastAPI: It is a modern Python web framework designed to: provide lightweight microframework with an intuitive, Flask-like routing system. ASGI server made FastAPI easy to run an async event loop that counts incoming requests.
- Docker: It will package the the module so that it can be run anywhere without any dependencies
- Github Actions: Used to push docker images to Docker Hub
- Docker Compose: It will help anyone to run the whole app with a single file without any additional dependencies