{

"stack": "python",

"packages": [

"flask",

"pytesseract",

"pillow",

"scikit-learn",

"openai"

],

"runCommand": "python main.py",

"openFiles": [

"main.py"

],

"files": {

"main.py":

"from flask import Flask, request, jsonify\n"

"import pytesseract\n"

"from PIL import Image\n"

"import io\n\n"

"app = Flask(\_\_name\_\_)\n\n"

"@app.route('/upload', methods=['POST'])\n"

"def upload\_document():\n"

" f = request.files['file']\n"

" img = Image.open(io.BytesIO(f.read()))\n"

" text = pytesseract.image\_to\_string(img)\n"

" label = classify(text)\n"

" errors = validate(label, text)\n"

" return jsonify({ 'text': text, 'label': label, 'validation\_errors': errors })\n\n"

"def classify(text):\n"

" tx = text.lower()\n"

" if 'kwh' in tx or 'meter reading' in tx:\n"

" return 'Electricity Bill'\n"

" if 'property tax' in tx:\n"

" return 'Property Tax Bill'\n"

" if 'birth certificate' in tx:\n"

" return 'Birth Certificate'\n"

" if 'mobile phone' in tx or 'call charges' in tx:\n"

" return 'Mobile Phone Bill'\n"

" return 'Unknown'\n\n"

"def validate(label, text):\n"

" errors = []\n"

" tx = text.lower()\n"

" if label == 'Electricity Bill' and 'due date' not in tx:\n"

" errors.append('Missing due date')\n"

" # add more rules here\n"

" return errors\n\n"

"if \_\_name\_\_ == '\_\_main\_\_':\n"

" app.run(host='0.0.0.0', port=3000)\n",

"model\_train.py":

"import csv\n"

"import pickle\n"

"from sklearn.feature\_extraction.text import TfidfVectorizer\n"

"from sklearn.linear\_model import LogisticRegression\n\n"

"# Load your training CSV: each row has 'text','label'\n"

"texts, labels = [], []\n"

"with open('training\_data.csv', newline='') as csvfile:\n"

" reader = csv.DictReader(csvfile)\n"

" for row in reader:\n"

" texts.append(row['text'])\n"

" labels.append(row['label'])\n\n"

"vect = TfidfVectorizer().fit(texts)\n"

"X = vect.transform(texts)\n"

"clf = LogisticRegression().fit(X, labels)\n\n"

"with open('model.pkl', 'wb') as f:\n"

" pickle.dump((vect, clf), f)\n",

"requirements.txt":

"Flask\n"

"pytesseract\n"

"Pillow\n"

"scikit-learn\n"

"openai\n",

"templates/index.html":

"<!DOCTYPE html>\n"

"<html lang=\"en\">\n"

"<head><meta charset=\"UTF-8\"><title>Doc Classifier</title></head>\n"

"<body>\n"

" <h1>Upload a Document</h1>\n"

" <form id=\"upload-form\" enctype=\"multipart/form-data\">\n"

" <input type=\"file\" name=\"file\" accept=\"image/\*,.pdf\" />\n"

" <button type=\"submit\">Classify</button>\n"

" </form>\n"

" <pre id=\"result\"></pre>\n"

" <script>\n"

" document.getElementById('upload-form').addEventListener('submit', async e => {\n"

" e.preventDefault();\n"

" const form = e.target;\n"

" const data = new FormData(form);\n"

" const res = await fetch('/upload', { method: 'POST', body: data });\n"

" document.getElementById('result').textContent = JSON.stringify(await res.json(), null, 2);\n"

" });\n"

" </script>\n"

"</body>\n"

"</html>"

},

"instructions": "Implement a simple Flask-based document classifier: OCR using pytesseract, rule-based classify(), validate() for required fields, and serve an HTML frontend under /templates. Once working, optionally train model\_train.py to produce model.pkl and integrate it into main.py."

}