import os

import json

import fitz  # PyMuPDF

from openai import OpenAI

from dotenv import load\_dotenv

load\_dotenv()

client = OpenAI(api\_key=os.getenv("OPENAI\_API\_KEY"))

def extract\_text\_from\_pdf(pdf\_path):

    """Extract full text from the given PDF."""

    doc = fitz.open(pdf\_path)

    text = ""

    for page in doc:

        text += page.get\_text()

    return text

def parse\_latest\_rate\_card(folder\_path="app/data/rate\_cards"):

    """Parse the latest rate card PDF file in the folder."""

    pdf\_files = [f for f in os.listdir(folder\_path) if f.endswith(".pdf")]

    if not pdf\_files:

        print("❌ No PDF files found in rate card folder.")

        return {}

    latest\_pdf = max(

        [os.path.join(folder\_path, f) for f in pdf\_files],

        key=os.path.getctime

    )

    print(f"🗂️ Parsing rate card from: {latest\_pdf}")

    text = extract\_text\_from\_pdf(latest\_pdf)

    # OpenAI Prompt

    prompt = f"""

You are a financial document parser. Extract the fixed mortgage rates from the following rate card text.

Return the output as JSON in this exact format:

{{

  "LVR <= 80%": {{

    "6 Months": 6.85,

    "1 Year": 6.99,

    "2 Years": 7.10,

    "3 Years": 7.25,

    "5 Years": 7.45

  }},

  "LVR > 80%": {{

    "6 Months": 7.05,

    "1 Year": 7.20,

    "2 Years": 7.30,

    "3 Years": 7.45,

    "5 Years": 7.65

  }}

}}

➕ Only include \*\*fixed rate terms\*\* like 6 months, 1 year, 2 years, 3 years, 5 years.

❌ Do not include floating rates, revolving rates, or variable rates.

The rates should be grouped under:

- "LVR <= 80%" for customers with loan-to-value ratio of 80% or less.

- "LVR > 80%" for customers with loan-to-value ratio greater than 80%.

Here is the rate card text:

\"\"\"{text}\"\"\"

"""

    try:

        response = client.chat.completions.create(

            model="gpt-4o",

            messages=[

                {"role": "user", "content": prompt}

            ],

            temperature=0

        )

        reply = response.choices[0].message.content

        print("🔍 OpenAI response:", reply)

        # Remove possible code block markdown ```json ... ```

        if reply.startswith("```json"):

            reply = reply.lstrip("```json").rstrip("```").strip()

        elif reply.startswith("```"):

            reply = reply.lstrip("```").rstrip("```").strip()

        rates = json.loads(reply)

    except Exception as e:

        print(f"❌ Error parsing rates: {e}")

        rates = {}

    print(f"📈 Extracted Rates: {rates}")

    return rates

# ✅ Run as standalone script to test

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

    rates = parse\_latest\_rate\_card()

    print("✅ Final Rates:", rates)