

# --------------------  extract-excel.py  (FULL FILE – 27 Apr 2025)  --------------------

import os, base64, json, re, io, imghdr, requests, pandas as pd, streamlit as st

API\_KEY = "AIzaSyDb8iBV1EWqLvjheG\_44gh3vQHfpmYGOCI"

from PIL import Image

from openpyxl import load\_workbook

def fill\_template\_from\_validated(validated\_path, template\_path,

                                 key\_col\_template='ASSETNUM',

                                 key\_col\_validated='Correct\_POWTR\_CODE',

                                 sheet\_name='AssetAttr') -> io.BytesIO:

    # 1) อ่าน df\_validate

    df = pd.read\_excel(validated\_path)

    df = df[df['Is\_Correct']==True].set\_index(key\_col\_validated)

    # 2) โหลด workbook + sheet

    wb = load\_workbook(template\_path)

    ws = wb[sheet\_name]

    # 3) ลบแถวแรก (group header)

    ws.delete\_rows(1)

    # 4) อ่าน header map: name→col\_index

    header = next(ws.iter\_rows(min\_row=1, max\_row=1))

    cols = {cell.value: cell.column for cell in header if cell.value}

    # 5) วนแต่ละแถว (จาก row 2) เติมข้อมูล

    for row in ws.iter\_rows(min\_row=2):

        asset = row[cols['ASSETNUM']-1].value

        if asset in df.index:

            rec = df.loc[asset]

            # ASSETNUM เป็น key ใหม่ (Correct\_POWTR\_CODE)

            row[cols['ASSETNUM']-1].value = rec.name

            # SITEID ← Plant, HIERARCHYPATH ← Location Description

            row[cols['SITEID']-1].value = rec['Plant']

            row[cols['HIERARCHYPATH']-1].value = rec['Location Description']

    # 6) เซฟเป็น BytesIO

    buf = io.BytesIO()

    wb.save(buf)

    buf.seek(0)

    return buf

# --------------------------------------------------------------------------- #

# 1)  Utilities                                                               #

# --------------------------------------------------------------------------- #

def encode\_image(file) -> tuple[str, str]:

    """Convert an uploaded image file to (base64‑string, mime‑type) for Gemini."""

    raw = file.getvalue()

    kind = imghdr.what(None, raw) or 'jpeg'

    mime = f"image/{kind}"

    return base64.b64encode(raw).decode('utf-8'), mime

def \_kv\_from\_text(txt: str) -> float | None:

    """

    Return the highest \*\*system voltage\*\* (kV) found in \*txt\*.

    • Accept “… 525000 V”, “34.5 kV”, “220 kV”, etc.

    • Ignore values that are clearly power/current ratings (kVA, A, kA, VA).

    • Skip numbers near “BIL” or “IMPULSE”.

    • Discard absurdly large values > 1500 kV.

    """

    txt\_u = txt.upper()

    best = None

    # split on '/', ',', ';' to isolate “33/30000/309000 kV” cases

    for chunk in re.split(r'[\/,;]', txt\_u):

        chunk = chunk.strip()

        # skip chunks that mention KVA / VA / KA / A

        if re.search(r'\bK?VA\b|\bKA\b|\bAMP|\bA\b', chunk):

            continue

        # skip if near BIL / IMPULSE

        if 'BIL' in chunk or 'IMPULSE' in chunk:

            continue

        for m in re.finditer(r'(\d+(?:\.\d+)?)\s\*([K]?V)(?![A-Z])', chunk):

            n = float(m.group(1))

            unit = m.group(2).upper()

            kv = n if unit == 'KV' else n / 1000

            if kv > 1500:          # absurdly high → ignore

                continue

            best = kv if best is None else max(best, kv)

    return best

# --------------------------------------------------------------------------- #

# 2)  Prompt generator                    #

# --------------------------------------------------------------------------- #

def generate\_prompt\_from\_excel(excel\_file):

    """

    Read an Excel list of attributes + (optionally) units, then build a Thai prompt

    telling Gemini to extract those exact fields in JSON.

    """

    # ----- read Excel whether it has a header row or not -----

    try:

        df = pd.read\_excel(excel\_file)

        first\_col = df.columns[0]

        is\_numeric\_header = isinstance(first\_col, (int, float))

        if is\_numeric\_header:

            excel\_file.seek(0)

            df = pd.read\_excel(excel\_file, header=None)

            df.columns = ['attribute\_name'] + [f'col\_{i}' for i in range(1, len(df.columns))]

            st.info("ตรวจพบไฟล์ไม่มีหัวคอลัมน์ – กำลังปรับให้อ่านได้")

    except Exception as e:

        excel\_file.seek(0)

        df = pd.read\_excel(excel\_file, header=None)

        df.columns = ['attribute\_name'] + [f'col\_{i}' for i in range(1, len(df.columns))]

        st.warning(f"อ่านไฟล์แบบมีหัวคอลัมน์ไม่ได้: {e}  → ใช้โหมดไม่มีหัว")

    st.write("คอลัมน์ที่พบ:", list(df.columns))

    attribute\_col = 'attribute\_name'

    if attribute\_col not in df.columns:

        for c in ['attribute\_name', 'attribute', 'name', 'attributes',

                  'Attribute', 'ATTRIBUTE', 'field', 'Field', 'FIELD']:

            if c in df.columns:

                attribute\_col = c; break

        if attribute\_col not in df.columns:

            attribute\_col = df.columns[0]

            st.warning(f"ไม่พบคอลัมน์ชื่อ attribute ที่รู้จัก – ใช้คอลัมน์ '{attribute\_col}' แทน")

    unit\_col = None

    for c in ['unit\_of\_measure', 'unit', 'Unit', 'UNIT', 'uom', 'UOM',

              'unit of measure', 'Unit of Measure']:

        if c in df.columns:

            unit\_col = c; break

    if unit\_col is None and len(df.columns) > 1:

        potential = df.columns[1]

        sample = df[potential].dropna().astype(str).tolist()[:10]

        if any(any(k in v for k in ['kg', 'V', 'A', 'kV', 'kVA', 'C', '°C',

                                    'mm', 'cm', 'm', '%']) for v in sample):

            unit\_col = potential

            st.info(f"ตรวจพบคอลัมน์ '{potential}' อาจเป็นหน่วยวัด")

    prompt\_parts = ["""กรุณาสกัดข้อมูลทั้งหมดจากรูปภาพนี้และแสดงผลในรูปแบบ JSON ที่มีโครงสร้างชัดเจน โดยใช้ key เป็นภาษาอังกฤษและ value เป็นข้อมูลที่พบ

ให้ return ค่า attributes กลับด้วยค่า attribute เท่านั้นห้าม return เป็น index เด็ดขาดและไม่ต้องเอาค่า index มาด้วย ให้ระวังเรื่อง voltage high side หน่วยต้องเป็น V หรือ kV เท่านั้น

โดยเอาเฉพาะ attributes ดังต่อไปนี้\n"""]

    for i, row in df.iterrows():

        attr = str(row[attribute\_col]).strip()

        if pd.isna(attr) or attr == '':

            continue

        if unit\_col and unit\_col in df.columns and pd.notna(row[unit\_col]) and str(row[unit\_col]).strip():

            prompt\_parts.append(f"{i+1}: {attr} [{row[unit\_col]}]")

        else:

            prompt\_parts.append(f"{i+1}: {attr}")

    prompt\_parts.append("\nหากไม่พบข้อมูลสำหรับ attribute ใด ให้ใส่ค่า - แทน ไม่ต้องเดาค่า และให้รวม attribute และหน่วยวัดไว้ในค่าที่ส่งกลับด้วย")

    return "\n".join(prompt\_parts)

# --------------------------------------------------------------------------- #

# 3)  Gemini API call                                                         #

# --------------------------------------------------------------------------- #

def extract\_data\_from\_image(api\_key: str, img\_b64: str, mime: str, prompt: str) -> str:

    url = f"https://generativelanguage.googleapis.com/v1beta/models/gemini-2.5-flash-preview-04-17:generateContent?key={api\_key}"

    payload = {

        "contents": [{

            "parts": [

                {"text": prompt},

                {"inlineData": {"mimeType": mime, "data": img\_b64}}

            ]

        }],

        "generationConfig": {"temperature": 0.2, "topP": 0.85, "maxOutputTokens": 9000}

    }

    r = requests.post(url, headers={"Content-Type": "application/json"}, data=json.dumps(payload))

    if r.ok and r.json().get('candidates'):

        return r.json()['candidates'][0]['content']['parts'][0]['text']

    return f"API ERROR {r.status\_code}: {r.text}"

# --------------------------------------------------------------------------- #

# 4)  POWTR‑CODE generator                                                    #

# --------------------------------------------------------------------------- #

def generate\_powtr\_code(extracted: dict) -> str:

    try:

# 1) Phase

        phase = '3'

        if any(any(k in str(v).upper() for k in ('1PH', '1-PH', 'SINGLE'))

               for v in extracted.values()):

            phase = '1'

        # 2) Voltage level

        high\_kv = None

        for k, v in extracted.items():

            if any(t in k.upper() for t in ('VOLT', 'HV', 'LV', 'RATED', 'SYSTEM')):

                kv = \_kv\_from\_text(str(v))

                if kv is not None:

                    high\_kv = kv if high\_kv is None else max(high\_kv, kv)

        if high\_kv is None:

            v\_char = '-'

        elif high\_kv > 765:

            return 'POWTR-3-OO'

        elif high\_kv >= 345:

            v\_char = 'E'

        elif high\_kv >= 100:

            v\_char = 'H'

        elif high\_kv >= 1:

            v\_char = 'M'

        else:

            v\_char = 'L'

        # 3) Type → default = '-'  (เมื่อตรวจไม่เจอทั้ง DRY และ OIL)

        t\_char = '-'

        for v in extracted.values():

            u = str(v).upper()

            if 'DRY' in u:

                t\_char = 'D'

                break

            # ตรวจหา oil-based cooling class (OIL, ONAN, OFAF, ...)

            if any(kw in u for kw in ('OIL', 'ONAN', 'OFAF', 'OA', 'FOA')):

                t\_char = 'O'

                break

        # 4) Tap‑changer (เหมือนเดิม)

        tap\_char = 'F'

        for v in extracted.values():

            u = str(v).upper()

            if any(x in u for x in ('ON‑LOAD', 'ON-LOAD', 'OLTC')):

                tap\_char = 'O'

                break

            if any(x in u for x in ('OFF‑LOAD', 'OFF-LOAD', 'FLTC', 'OCTC')):

                tap\_char = 'F'

        code = f'POWTR-{phase}{v\_char}{t\_char}{tap\_char}'

        prefix = code.split('-',1)[0]   # จะได้ 'POWTR'

        return f"{prefix} \\ {code}"

    except Exception:

        return 'ไม่สามารถระบุได้'

def add\_powtr\_codes(results):

    for r in results:

        d = r.get('extracted\_data', {})

        if isinstance(d, dict) and not any(k in d for k in ('error','raw\_text')):

            # ใส่ POWTR\_CODE เข้าไป

            code = generate\_powtr\_code(d)

            d['POWTR\_CODE'] = code

    return results

def split\_value\_unit(raw: object) -> tuple[str, str]:

    """

    แยก raw string เป็น (value, unit)

    - ถ้า raw เริ่มด้วยตัวเลข (อาจมี - และทศนิยม) ให้จับตัวเลขมาเป็น value ส่วนที่เหลือเป็น unit

    - กรณีอื่น ๆ (text, dash, slash) ให้ value = raw, unit = ''

    """

    s = str(raw or "").strip()

    # 1) กรณีขึ้นต้นด้วยตัวเลข (optional - และ .)

    m = re.match(r'^(-?\d+(?:\.\d+)?)(.\*)$', s)

    if m:

        val = m.group(1)

        unit = m.group(2).strip()

        # ปฏิเสธ unit ที่มี '/' หรือ '-'

        if '/' in unit or '-' in unit:

            return s, ''

        return val, unit

    # 2) กรณีเปอร์เซนต์ติดหลัง (เช่น "13.46%")

    m2 = re.match(r'^(-?\d+(?:\.\d+)?)\s\*%$', s)

    if m2:

        return m2.group(1), '%'

    # 3) กรณีมีหน่วยที่เป็นตัวอักษรหรือตัวสัญลักษณ์ (°C, A, V, kg ...)

    m3 = re.match(r'^(-?\d+(?:\.\d+)?)\s\*([°%A-Za-zµΩ]+)$', s)

    if m3:

        return m3.group(1), m3.group(2)

    # 4) กรณีอื่น ๆ (text, dash, slash) เอา raw ทั้งหมดเป็น VALUE, unit = ''

    return s, ''

 #--------------------------------------------------------------------------- #

# 5)  Streamlit UI

# ---------------------------

# Prompt เริ่มต้น เมื่อ user ไม่โหลดไฟล์ Excel มา

# … import ทั้งหมดเหมือนเดิม …

# Prompt เริ่มต้น (ต้องประกาศไว้ก่อน UI)

default\_prompt = """กรุณาสกัดข้อมูลทั้งหมดจากรูปภาพนี้และแสดงผลในรูปแบบ JSON …"""

tab1, tab2 = st.tabs(["สกัดจากรูปภาพ","ประมวลผลจาก validated"])

# ------ โหมด 1: สกัดจากรูปภาพ ------

with tab1:

    st.subheader("💡 สกัดข้อมูลจากรูปภาพ")

    excel\_f1 = st.file\_uploader("1. เลือกไฟล์ Excel attributes", ["xlsx","xls"], key="tab1\_attr")

    imgs    = st.file\_uploader("2. อัปโหลดรูปภาพ (หลายไฟล์)", ["jpg","png","jpeg"],

                               accept\_multiple\_files=True, key="tab1\_imgs")

    if st.button("ประมวลผลภาพ", key="btn\_extract") and excel\_f1 and imgs:

        # --- เอาโค้ดบล็อก extraction+wide มาใส่ที่นี่ ---

        prompt = default\_prompt

        # ถ้ามี excel\_f1 ให้ generate\_prompt\_from\_excel…

        if excel\_f1:

            prompt = generate\_prompt\_from\_excel(excel\_f1)

        st.expander("Prompt").write(prompt)

        # สกัดรูป → results …

        results, bar, status = [], st.progress(0), st.empty()

        for i,f in enumerate(imgs,1):

            bar.progress(i/len(imgs))

            status.write(f"กำลังประมวลผล {i}/{len(imgs)} – {f.name}")

            b64,mime = encode\_image(f)

            js = {}

            resp = extract\_data\_from\_image(API\_KEY, b64, mime, prompt)

            try:

                js = json.loads(resp[resp.find('{'):resp.rfind('}')+1])

            except:

                js = {"error":resp}

            results.append({"file\_name":f.name,"extracted\_data":js})

        # เติม POWTR\_CODE

        results = add\_powtr\_codes(results)

        # สร้าง df\_long, pivot→df\_wide, แสดง + ดาวน์โหลด wide

        rows = []

        for r in results:

            d = r["extracted\_data"]

            assetnum = d.get("ASSETNUM","")

            siteid   = d.get("SITEID","")

            powtr    = d.get("POWTR\_CODE","")

            if "error" in d or "raw\_text" in d:

                rows.append({

                    "ASSETNUM":assetnum,

                    "SITEID":siteid,

                    "POWTR\_CODE":powtr,

                    "ATTRIBUTE":"Error",

                    "VALUE":d.get("error",d.get("raw\_text",""))

                })

            else:

                for attr,val in d.items():

                    if attr in ("ASSETNUM","SITEID","POWTR\_CODE"): continue

                    rows.append({

                        "ASSETNUM":assetnum,

                        "SITEID":siteid,

                        "POWTR\_CODE":powtr,

                        "ATTRIBUTE":attr,

                        "VALUE":val

                    })

        df\_long = pd.DataFrame(rows)

        st.subheader("ตัวอย่างข้อมูล (แบบแถว)")

        st.dataframe(df\_long)

        buf = io.BytesIO()

        with pd.ExcelWriter(buf, engine="openpyxl") as w:

            df\_long.to\_excel(w, index=False)

        buf.seek(0)

        st.download\_button("ดาวน์โหลด long", buf, "extracted\_long.xlsx")

# ------ โหมด 2: ประมวลผลจาก validated ------

# ------ โหมด 2: ประมวลผลจาก validated ------

with tab2:

    st.header("🔍 ประมวลผลจากไฟล์ validated")

    # 1) Uploader

    excel\_attr2    = st.file\_uploader("1) เลือกไฟล์ Excel Attributes", ["xlsx","xls"], key="tab2\_attr")

    validated\_file = st.file\_uploader("2) เลือกไฟล์ validated\_powtr\_codes.xlsx", ["xlsx"], key="tab2\_val")

    if st.button("ประมวลผล validated", key="btn\_valid"):

        if not excel\_attr2 or not validated\_file:

            st.warning("กรุณาอัปโหลดทั้งไฟล์ Attributes และ validated")

            st.stop()

        # 2) โหลด canonical attributes

        df\_attr   = pd.read\_excel(excel\_attr2)

        # ดึงเฉพาะคอลัมน์แรกเป็น list ของชื่อ attribute

        canonical = df\_attr[df\_attr.columns[0]].dropna().astype(str).tolist()

        # 3) โหลดและกรอง validated

        df\_val = pd.read\_excel(validated\_file)

        df\_val = df\_val[df\_val["Is\_Correct"] == True]

        st.subheader("Filtered validated (Is\_Correct == True)")

        st.dataframe(df\_val)

        # 4) สร้างตาราง long พร้อมเว้นบรรทัดว่างระหว่างแต่ละ transformer

        rows = []

        prev\_asset = None

        for \_, row in df\_val.iterrows():

            # ดึง ASSETNUM และ SITEID ตามที่ต้องการ

            assetnum = row.get("Location", "")        # ถ้าในไฟล์เป็นคอลัมน์อื่น ให้แก้ชื่อที่นี่

            plant    = row.get("Plant", "")

            siteid   = (plant[:3] + "0") if plant else ""

            powtr    = row.get("Correct\_POWTR\_CODE", "")

            for attr in canonical:

                raw = row.get(attr, "-")

                # ← ตรงนี้ครับ ให้เช็คก่อนว่าเป็น Serial No.

                if attr.strip().lower().startswith("serial"):

                    val  = str(raw).strip()

                    unit = ""

                else:

                    val, unit = split\_value\_unit(raw)

            # ถ้าเจอ transformer ใหม่ ให้แทรก row ว่าง

            if prev\_asset is not None and assetnum != prev\_asset:

                rows.append({

                    "ASSETNUM":"", "SITEID":"", "POWTR\_CODE":"",

                    "ATTRIBUTE":"", "VALUE":"", "MEASUREUNIT":""

                })

            prev\_asset = assetnum

            # วน attribute ตาม canonical list

            for attr in canonical:

                raw = row.get(attr, "-")  # ถ้าไม่มีคอลัมน์ ก็แสดง '-'

                val, unit = split\_value\_unit(raw)

                rows.append({

                    "ASSETNUM":    assetnum,

                    "SITEID":      siteid,

                    "POWTR\_CODE":  powtr,

                    "ATTRIBUTE":   attr,

                    "VALUE":       val,

                    "MEASUREUNIT": unit

                })

        df\_long\_val = pd.DataFrame(rows)

        st.subheader("ผลลัพธ์แบบแถว (long) จาก validated")

        st.dataframe(df\_long\_val)

        # 5) ดาวน์โหลดไฟล์ long

        buf = io.BytesIO()

        with pd.ExcelWriter(buf, engine="openpyxl") as writer:

            df\_long\_val.to\_excel(writer, index=False)

        buf.seek(0)

        st.download\_button(

            "ดาวน์โหลด extracted\_long\_from\_validated.xlsx",

            buf,

            "extracted\_long\_from\_validated.xlsx",

            key="dl\_long\_val"

        )

