# 🚀 SKF TC Generator - Udit generated this code for CS team

!pip install pandas openpyxl python-docx > /dev/null 2>&1

!apt-get update > /dev/null 2>&1 && apt-get install -y libreoffice > /dev/null 2>&1

import pandas as pd

from docx import Document

from docx.shared import Inches

from docx.oxml import OxmlElement

from docx.oxml.ns import qn

import os, re, zipfile, subprocess

from datetime import datetime

def convert\_to\_pdf(word\_file, output\_folder):

"""🔄 Fast PDF conversion"""

try:

base\_name = os.path.splitext(os.path.basename(word\_file))[0]

cmd = ['libreoffice', '--headless', '--convert-to', 'pdf', '--outdir', output\_folder, word\_file]

subprocess.run(cmd, capture\_output=True, timeout=30)

pdf\_file = os.path.join(output\_folder, f"{base\_name}.pdf")

if os.path.exists(pdf\_file):

os.remove(word\_file) # Clean up

return pdf\_file

return word\_file

except:

return word\_file

def upload\_files():

"""📁 Quick file upload"""

print("🎯 SKF TC Generator - Udit generated this code for CS team")

print("📊 Step 1: Excel file")

from google.colab import files as colab\_files

excel\_files = colab\_files.upload()

excel\_file = list(excel\_files.keys())[0] if excel\_files else None

print("📄 Step 2: TC Template")

word\_files = colab\_files.upload()

word\_file = list(word\_files.keys())[0] if word\_files else None

return excel\_file, word\_file

def read\_data(excel\_file):

"""📊 Smart data reading"""

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

df = df.astype(str).replace('nan', 'N/A')

# Find invoice column

invoice\_col = None

for col in df.columns:

if any(x in col.lower() for x in ['invoice', 'inv']):

invoice\_col = col

break

if not invoice\_col:

print("❌ No invoice column found!")

return None, None

# Group by invoice

grouped = {}

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

inv = str(row[invoice\_col]).strip()

if inv not in grouped:

grouped[inv] = []

grouped[inv].append(row.to\_dict())

print(f"✅ Loaded: {len(df)} records → {len(grouped)} invoices")

return df, grouped

def create\_beautiful\_table(doc, table\_data):

"""✨ Create beautiful table with borders"""

table = doc.add\_table(rows=1, cols=3)

# Add borders

def add\_borders(tbl):

try:

tbl\_element = tbl.\_element

tbl\_pr = tbl\_element.tblPr

borders = OxmlElement('w:tblBorders')

for border\_name in ['top', 'left', 'bottom', 'right', 'insideH', 'insideV']:

border = OxmlElement(f'w:{border\_name}')

border.set(qn('w:val'), 'single')

border.set(qn('w:sz'), '6')

border.set(qn('w:color'), '000000')

borders.append(border)

tbl\_pr.append(borders)

except:

pass

add\_borders(table)

# Set column widths

table.columns[0].width = Inches(0.8)

table.columns[1].width = Inches(3.2)

table.columns[2].width = Inches(1.0)

# Headers

headers = ["Sr. No.", "SKF Product Designation", "Quantity Supplied"]

for i, header in enumerate(headers):

cell = table.rows[0].cells[i]

cell.text = header

for paragraph in cell.paragraphs:

paragraph.alignment = 1 # Center

for run in paragraph.runs:

run.bold = True

# Data rows

for row\_data in table\_data:

row = table.add\_row()

for i, data in enumerate(row\_data):

cell = row.cells[i]

cell.text = str(data)

if i in [0, 2]: # Center Sr.No and Quantity

cell.paragraphs[0].alignment = 1

return table

def fill\_template(template\_path, invoice\_data, invoice\_number):

"""🔥 Fast template filling"""

doc = Document(template\_path)

first\_record = invoice\_data[0]

# Get data from specific columns

cols = list(first\_record.keys())

ship\_to = first\_record.get(cols[19] if len(cols) > 19 else '', 'N/A')

# Get Customer Reference from column G (index 6, 0-based)

customer\_reference = first\_record.get(cols[6] if len(cols) > 6 else '', 'N/A')

# Get Customer Postal Code from column BK (index 62, 0-based - BK is the 63rd column)

customer\_postal\_code = first\_record.get(cols[62] if len(cols) > 62 else '', 'N/A')

# Quick replacements

replacements = {

'[Invoice Date]': first\_record.get('Invoice Date', 'N/A'),

'[Local invoice Number]': str(invoice\_number),

'[Ship-To-Party Name]': ship\_to,

'[Address Line 1]': first\_record.get('Address Line 1', 'N/A'),

'[Address Line 2]': first\_record.get('Address Line 2', 'N/A'),

'[Address Line 3]': customer\_postal\_code, # Now using Customer Postal Code from column BK

'[Customer Postal Code]': customer\_postal\_code, # Adding both placeholders for flexibility

'[Customer No]': customer\_reference, # Now using Customer Reference from column G

'[Customer Reference]': customer\_reference # Adding both placeholders for flexibility

}

# Replace in all paragraphs

for paragraph in doc.paragraphs:

for old, new in replacements.items():

if old in paragraph.text:

paragraph.text = paragraph.text.replace(old, str(new))

# Replace in tables

for table in doc.tables:

for row in table.rows:

for cell in row.cells:

for paragraph in cell.paragraphs:

for old, new in replacements.items():

if old in paragraph.text:

paragraph.text = paragraph.text.replace(old, str(new))

# Create product table data

table\_data = []

for i, record in enumerate(invoice\_data, 1):

cols = list(record.keys())

skf\_product = record.get(cols[8] if len(cols) > 8 else '', 'N/A')

quantity = record.get(cols[9] if len(cols) > 9 else '', 'N/A')

table\_data.append([str(i), skf\_product, quantity])

# Find "Dear Sir," and insert table before it

for i, paragraph in enumerate(doc.paragraphs):

if "Dear Sir," in paragraph.text:

# Create beautiful table

new\_table = create\_beautiful\_table(doc, table\_data)

# Insert before Dear Sir

p = paragraph.\_element

table\_element = new\_table.\_element

p.addprevious(table\_element)

break

return doc

def generate\_certificates(template\_path, grouped\_data):

"""⚡ Lightning fast certificate generation"""

output\_folder = "TC\_Certificates"

os.makedirs(output\_folder, exist\_ok=True)

generated = []

total = len(grouped\_data)

print(f"🚀 Generating {total} certificates...")

for i, (invoice\_num, records) in enumerate(grouped\_data.items(), 1):

try:

# Fill template

doc = fill\_template(template\_path, records, invoice\_num)

# Save

clean\_name = re.sub(r'[<>:"/\\|?\*\s]', '\_', str(invoice\_num))

word\_file = os.path.join(output\_folder, f"TC\_{clean\_name}.docx")

doc.save(word\_file)

# Convert to PDF

pdf\_file = convert\_to\_pdf(word\_file, output\_folder)

generated.append(pdf\_file)

# Progress

if i % 10 == 0 or i == total:

print(f" ✅ {i}/{total} completed")

except Exception as e:

print(f" ❌ Error on {invoice\_num}: {e}")

return generated

def create\_download():

"""📦 Create download package"""

timestamp = datetime.now().strftime("%Y%m%d\_%H%M")

zip\_name = f"TC\_Certificates\_{timestamp}.zip"

try:

with zipfile.ZipFile(zip\_name, 'w') as zipf:

for root, dirs, file\_list in os.walk("TC\_Certificates"):

for file in file\_list:

file\_path = os.path.join(root, file)

zipf.write(file\_path, file)

# Import files here to avoid conflicts

from google.colab import files as colab\_files

colab\_files.download(zip\_name)

print(f"✅ Downloaded: {zip\_name}")

except Exception as e:

print(f"❌ Download error: {e}")

print(f"📁 Files saved in: TC\_Certificates/ folder")

def main():

"""🎯 Main execution - Udit generated this code for CS team"""

print("🎯 SKF TC Generator - Udit generated this code for CS team")

print("=" \* 60)

# Upload

excel\_file, word\_file = upload\_files()

if not excel\_file or not word\_file:

return

# Read data

df, grouped\_data = read\_data(excel\_file)

if not df.empty and grouped\_data:

# Generate

generated = generate\_certificates(word\_file, grouped\_data)

# Download

if generated:

pdf\_count = len([f for f in generated if f.endswith('.pdf')])

print(f"\n🎉 Success! Generated {len(generated)} certificates")

print(f"📄 PDFs: {pdf\_count} | 📝 Word: {len(generated) - pdf\_count}")

create\_download()

else:

print("❌ No certificates generated")

# Run it!

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

main()