import pandas as pd

import PyPDF2

from reportlab.lib.pagesizes import letter, A4

from reportlab.platypus import SimpleDocTemplate, Table, TableStyle, Paragraph, Spacer, Image

from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle

from reportlab.lib.units import inch

from reportlab.lib import colors

from reportlab.lib.colors import HexColor

from reportlab.lib.enums import TA\_CENTER, TA\_LEFT

from datetime import datetime

import os

# File paths

base\_path = r"C:\Users\USER\Desktop\NEW STATEMENT"

csv\_file = os.path.join(base\_path, "Statements.csv")

template\_pdf = os.path.join(base\_path, "STATEMENT.pdf")

logo\_file = os.path.join(base\_path, "logo.jpeg")

output\_pdf = os.path.join(base\_path, "Generated\_Statement.pdf")

def extract\_header\_info(pdf\_path):

"""Extract header information from template PDF"""

try:

with open(pdf\_path, 'rb') as file:

reader = PyPDF2.PdfReader(file)

text = reader.pages[0].extract\_text()

# Extract company info (you can modify this based on your needs)

header\_info = {

'scheme\_name': 'BEST PENSION MASTER TRUST SCHEME',

'statement\_type': 'STANDARD PENSION TRUST STATEMENT OF ACCOUNT',

'employer\_name': 'ST. JOHN\'S HOSPITAL & FERTILITY CENTRE',

'employer\_address': 'N/A',

'contact\_number': '0204693533',

'issue\_date': datetime.now().strftime('%d-%b-%Y').upper()

}

return header\_info

except Exception as e:

print(f"Error reading PDF: {e}")

# Return default values if PDF reading fails

return {

'scheme\_name': 'BEST PENSION MASTER TRUST SCHEME',

'statement\_type': 'STANDARD PENSION TRUST STATEMENT OF ACCOUNT',

'employer\_name': 'ST. JOHN\'S HOSPITAL & FERTILITY CENTRE',

'employer\_address': 'N/A',

'contact\_number': '0204693533',

'issue\_date': datetime.now().strftime('%d-%b-%Y').upper()

}

def process\_csv\_data(csv\_path):

"""Process CSV data and group by year"""

try:

# Read CSV

df = pd.read\_csv(csv\_path)

# Convert Value Date to datetime

df['Value Date'] = pd.to\_datetime(df['Value Date'])

# Extract year

df['Year'] = df['Value Date'].dt.year

# Format date for display

df['Formatted\_Date'] = df['Value Date'].dt.strftime('%d-%b-%y').str.upper()

# Clean up Details column - create description

df['Description'] = df['Details'].apply(lambda x: f"Payment received on {x.split(',')[0]}" if pd.notna(x) and x.strip() else "Payment received")

# Group by year and sort

grouped\_data = {}

for year in sorted(df['Year'].unique()):

year\_data = df[df['Year'] == year].copy()

year\_data = year\_data.sort\_values('Value Date')

grouped\_data[year] = year\_data[['Formatted\_Date', 'Description', 'Amount']].values.tolist()

return grouped\_data

except Exception as e:

print(f"Error processing CSV: {e}")

return {}

def create\_pdf\_statement(header\_info, grouped\_data, output\_path):

"""Create PDF statement with the template format"""

try:

# Create document

doc = SimpleDocTemplate(output\_path, pagesize=A4,

rightMargin=0.5\*inch, leftMargin=0.5\*inch,

topMargin=0.5\*inch, bottomMargin=0.5\*inch)

# Get styles

styles = getSampleStyleSheet()

# Create custom styles

title\_style = ParagraphStyle(

'CustomTitle',

parent=styles['Heading1'],

fontSize=14,

spaceAfter=6,

alignment=TA\_CENTER,

fontName='Helvetica-Bold'

)

header\_style = ParagraphStyle(

'CustomHeader',

parent=styles['Normal'],

fontSize=11,

spaceAfter=4,

alignment=TA\_LEFT,

fontName='Helvetica-Bold'

)

normal\_style = ParagraphStyle(

'CustomNormal',

parent=styles['Normal'],

fontSize=10,

spaceAfter=4,

alignment=TA\_LEFT

)

year\_style = ParagraphStyle(

'YearStyle',

parent=styles['Normal'],

fontSize=11,

spaceAfter=6,

spaceBefore=12,

alignment=TA\_LEFT,

fontName='Helvetica-Bold'

)

# Story (content) list

story = []

# Add logo at the top left

try:

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

logo = Image(logo\_file)

logo.drawHeight = 1.5\*inch # Adjust height as needed

logo.drawWidth = 2.5\*inch # Adjust width as needed

logo.hAlign = 'LEFT'

story.append(logo)

story.append(Spacer(1, 12))

print("Logo added successfully!")

else:

print(f"Logo file not found at: {logo\_file}")

except Exception as e:

print(f"Error loading logo: {e}")

# Continue without logo if file not found

# Add title

story.append(Paragraph(header\_info['scheme\_name'], title\_style))

story.append(Paragraph(header\_info['statement\_type'], title\_style))

story.append(Spacer(1, 12))

# Define colors to match the original

red\_color = HexColor('#CC3A3A') # Red color for headers and lines

gray\_color = HexColor('#CCCCCC') # Gray color for year headers

# Create red header style for EMPLOYER INFORMATION

employer\_header\_style = ParagraphStyle(

'EmployerHeader',

parent=styles['Normal'],

fontSize=11,

spaceAfter=8,

spaceBefore=4,

alignment=TA\_LEFT,

fontName='Helvetica-Bold',

textColor=colors.white,

backColor=red\_color,

leftIndent=6,

rightIndent=6,

)

# Add employer information with red header

story.append(Paragraph("EMPLOYER INFORMATION", employer\_header\_style))

story.append(Paragraph(f"NAME : {header\_info['employer\_name']}", normal\_style))

story.append(Paragraph(f"ADDRESS : {header\_info['employer\_address']}", normal\_style))

story.append(Paragraph(f"CONTACT NUMBER : {header\_info['contact\_number']}", normal\_style))

story.append(Paragraph(f"ISSUE DATE : {header\_info['issue\_date']}", normal\_style))

story.append(Spacer(1, 20))

# Add transaction data by year

for year in sorted(grouped\_data.keys()):

# Year header with gray background

year\_header\_style = ParagraphStyle(

'YearHeaderStyle',

parent=styles['Normal'],

fontSize=11,

spaceAfter=8,

spaceBefore=12,

alignment=TA\_CENTER,

fontName='Helvetica-Bold',

backColor=gray\_color,

leftIndent=6,

rightIndent=6

)

story.append(Paragraph(f"Year: {year}", year\_header\_style))

# Create table data

table\_data = [['Date', 'Description', 'Allocation']] # Header row

for transaction in grouped\_data[year]:

date\_str = transaction[0]

description = f"Payment received on {date\_str}"

amount = f"{float(transaction[2]):,.2f}"

table\_data.append([date\_str, description, amount])

# Create table with red header lines

table = Table(table\_data, colWidths=[1.2\*inch, 4\*inch, 1.3\*inch])

table.setStyle(TableStyle([

('ALIGN', (0, 0), (-1, -1), 'LEFT'),

('ALIGN', (2, 0), (2, -1), 'RIGHT'), # Right align amounts

('FONTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'),

('FONTSIZE', (0, 0), (-1, 0), 10),

('FONTNAME', (0, 1), (-1, -1), 'Helvetica'),

('FONTSIZE', (0, 1), (-1, -1), 9),

('BOTTOMPADDING', (0, 0), (-1, 0), 8),

('TOPPADDING', (0, 0), (-1, 0), 8),

('LINEBELOW', (0, 0), (-1, 0), 2, red\_color), # Red line under headers

('VALIGN', (0, 0), (-1, -1), 'MIDDLE'),

]))

story.append(table)

story.append(Spacer(1, 15))

# Add disclaimer

disclaimer\_style = ParagraphStyle(

'Disclaimer',

parent=styles['Normal'],

fontSize=8,

spaceAfter=4,

alignment=TA\_LEFT,

fontName='Helvetica-Oblique'

)

story.append(Spacer(1, 20))

story.append(Paragraph("Disclaimer: The information provided is provisional and subject to change based on further review and analysis. It should not be considered final or conclusive", disclaimer\_style))

# Build PDF

doc.build(story)

print(f"PDF statement generated successfully: {output\_path}")

except Exception as e:

print(f"Error creating PDF: {e}")

def main():

"""Main function to generate the statement"""

print("Starting PDF statement generation...")

# Extract header information

print("Extracting header information...")

header\_info = extract\_header\_info(template\_pdf)

# Process CSV data

print("Processing CSV data...")

grouped\_data = process\_csv\_data(csv\_file)

if not grouped\_data:

print("No data found in CSV file.")

return

print(f"Found data for years: {sorted(grouped\_data.keys())}")

# Create PDF

print("Creating PDF statement...")

create\_pdf\_statement(header\_info, grouped\_data, output\_pdf)

print("Process completed!")

# Run the main function

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

main()