from flask import Flask, render\_template, request, send\_file, jsonify
import pandas as pd
import os
import numpy as np
import logging

# Настройка логирования
logging.basicConfig(level=logging.DEBUG)
logger = logging.getLogger(\_\_name\_\_)

app = Flask(\_\_name\_\_)
app.config['UPLOAD\_FOLDER'] = 'uploads/'
# Создаем папку для загрузок, если её нет
if not os.path.exists(app.config['UPLOAD\_FOLDER']):
 os.makedirs(app.config['UPLOAD\_FOLDER'])
 logger.debug(f"Создана папка для загрузок: {app.config['UPLOAD\_FOLDER']}")

# Загружаем БД при старте приложения
db\_path = 'uploads/БД.xlsx' # Укажите путь к вашему файлу БД
try:
 df\_db = pd.read\_excel(db\_path, sheet\_name='БД')
 logger.debug("База данных успешно загружена")
except Exception as e:
 logger.error(f"Ошибка при загрузке БД: {e}")
 raise
# Проверка наличия столбца 'Тип'
required\_columns = ['Тип', 'Подтип', 'лямбда б', 'Кэ1', 'Кэ2', 'Кпр']
for column in required\_columns:
 if column not in df\_db.columns:
 logger.error(f"Столбец '{column}' отсутствует в файле БД. Добавьте его.")
 raise ValueError(f"Столбец '{column}' отсутствует в файле БД. Добавьте его.")

# Словарь с формулами и коэффициентами
formulas = {
 'Резистор': {
 'Подтип1': {'kt\_coef': 0.15, 'kn\_coef': 0.1234},
 'Подтип2': {'kt\_coef': 0.20, 'kn\_coef': 0.1500},
 'default': {'kt\_coef': 0.21, 'kn\_coef': 0.2341},
 },
 'Конденсатор': {
 'Подтип1': {'kt\_coef': 0.21, 'kn\_coef': 0.2341},
 'Подтип2': {'kt\_coef': 0.25, 'kn\_coef': 0.3000},
 'default': {'kt\_coef': 0.21, 'kn\_coef': 0.2341},
 },
 'Микросхема': {
 'Подтип1': {'kt\_coef': 0.10, 'kn\_coef': 0.1000},
 'default': {'kt\_coef': 0.10, 'kn\_coef': 0.1000},
 },
 'Диоды': {
 'Подтип1': {'kt\_coef': 0.12, 'kn\_coef': 0.1100},
 'default': {'kt\_coef': 0.12, 'kn\_coef': 0.1100},
 },
 'Транзисторы': {
 'Подтип1': {'kt\_coef': 0.18, 'kn\_coef': 0.2000},
 'default': {'kt\_coef': 0.18, 'kn\_coef': 0.2000},
 },
 'Прочие': {
 'default': {'kt\_coef': 0.21, 'kn\_coef': 0.2341},
 }
}

def calculate\_kt\_kn(row, r\_r, temperature, overheat\_temperature):
 element\_type = row['Тип']
 subtype = row.get('Подтип', 'default')
 coefficients = formulas.get(element\_type, {}).get(subtype, formulas['Прочие']['default'])
 kt = coefficients['kt\_coef'] \* r\_r \* (1 + overheat\_temperature / 100) # Пример формулы
 kn = coefficients['kn\_coef'] \* np.sqrt(2) \* temperature
 return kt, kn

@app.route('/')
def index():
 logger.debug("Отображение главной страницы")
 return render\_template('index.html')

@app.route('/upload', methods=['POST'])
def upload():
 try:
 logger.debug("Начало обработки запроса /upload")

 # Проверка наличия файла
 if 'file1' not in request.files:
 logger.error("Файл не загружен")
 return jsonify({"error": "Файл не загружен"}), 400
 file1 = request.files['file1']
 if file1.filename == '':
 logger.error("Файл не выбран")
 return jsonify({"error": "Файл не выбран"}), 400
 # Получаем данные из формы
 temperature = float(request.form['temperature'])
 r\_r = float(request.form['r\_r'])
 overheat\_temperature = float(request.form['overheat\_temperature']) # Новое поле
 k\_e = int(request.form['k\_e'])
 logger.debug(f"Получены данные: temperature={temperature}, r\_r={r\_r}, overheat\_temperature={overheat\_temperature}, k\_e={k\_e}")

 # Сохраняем файл
 file1\_path = os.path.join(app.config['UPLOAD\_FOLDER'], file1.filename)
 file1.save(file1\_path)
 logger.debug(f"Файл сохранен: {file1\_path}")

 # Чтение данных из файла
 df\_initial = pd.read\_excel(file1\_path, sheet\_name='Исходные данные')
 logger.debug("Исходные данные загружены")

 # Проверка наличия необходимых столбцов в загруженном файле
 required\_columns\_initial = ['id'] # Добавьте другие необходимые столбцы
 for column in required\_columns\_initial:
 if column not in df\_initial.columns:
 logger.error(f"Столбец '{column}' отсутствует в загруженном файле.")
 return jsonify({"error": f"Столбец '{column}' отсутствует в загруженном файле."}), 400
 # Выбор столбца Кэ1 или Кэ2
 k\_e\_column = 'Кэ1' if k\_e == 1 else 'Кэ2'
 df\_db['выбранный\_Кэ'] = df\_db[k\_e\_column]
 logger.debug(f"Выбран столбец: {k\_e\_column}")

 # Расчет коэффициентов Кт и Кн
 df\_db[['Кт', 'Кн']] = df\_db.apply(
 lambda row: pd.Series(calculate\_kt\_kn(row, r\_r, temperature, overheat\_temperature)), # Передаем overheat\_temperature
 axis=1
 )
 logger.debug("Коэффициенты Кт и Кн рассчитаны")

 # Расчет результата
 df\_db['Результат'] = (
 df\_db['лямбда б'] \*
 df\_db['выбранный\_Кэ'] \*
 df\_db['Кпр'] \*
 df\_db['Кт'] \*
 df\_db['Кн']
 )
 logger.debug("Результат рассчитан")

 # Объединение данных
 df\_result = pd.merge(df\_initial, df\_db, on='id', how='left')
 logger.debug("Данные объединены")

 # Сумма по типам
 sum\_by\_type = df\_result.groupby('Тип\_y')['Результат'].sum().reset\_index()
 sum\_by\_type = sum\_by\_type.rename(columns={'Результат': 'sum'})
 print(sum\_by\_type)
 total\_sum = df\_result['Результат'].sum()
 logger.debug(f"Сумма по типам: {sum\_by\_type}")
 logger.debug(f"Общая сумма: {total\_sum}")

 # Сохранение результата
 result\_path = os.path.join(app.config['UPLOAD\_FOLDER'], 'result.xlsx')
 df\_result.to\_excel(result\_path, index=False)
 logger.debug(f"Результат сохранен в {result\_path}")

 # Возвращаем результаты в формате JSON
 return jsonify({
 "results": [{"type": row['Тип\_y'], "sum": row['sum']} for \_, row in sum\_by\_type.iterrows()],
 "total\_sum": total\_sum
 })

 except Exception as e:
 logger.error(f"Ошибка: {e}", exc\_info=True)
 return jsonify({"error": str(e)}), 500
@app.route('/download')
def download():
 try:
 result\_path = os.path.join(app.config['UPLOAD\_FOLDER'], 'result.xlsx')
 logger.debug(f"Скачивание файла: {result\_path}")
 return send\_file(result\_path, as\_attachment=True)
 except Exception as e:
 logger.error(f"Ошибка при скачивании файла: {e}")
 return jsonify({"error": str(e)}), 500
if \_\_name\_\_ == '\_\_main\_\_':
 logger.debug("Запуск Flask-приложения")
 app.run(debug=True)

<!DOCTYPE html>
<html lang="ru">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Расчет интенсивности отказов</title>
 <link rel="stylesheet" href="{{ url\_for('static', filename='styles.css') }}">
</head>
<body>
 <div class="form-container">
 <h1>Расчет интенсивности отказов</h1>
 <form id="uploadForm" action="/upload" method="post" enctype="multipart/form-data">
 <label for="file1">Загрузите файл с исходными данными:</label>
 <input type="file" name="file1" id="file1" required>
 <label for="temperature">Температура:</label>
 <input type="number" step="0.01" name="temperature" id="temperature" required>
 <label for="overheat\_temperature">Температура перегрева:</label>
 <input type="number" step="0.01" name="overheat\_temperature" id="overheat\_temperature" required>
 <label for="r\_r">R<sub>r</sub>:</label>
 <input type="number" step="0.01" name="r\_r" id="r\_r" required>
 <label for="k\_e">Кэ:</label>
 <select name="k\_e" id="k\_e" required>
 <option value="1">Кэ1</option>
 <option value="2">Кэ2</option>
 </select>
 <button type="submit" id="calculateBtn">Рассчитать</button>
 <button type="button" id="downloadBtn" class="hidden" onclick="downloadExcel()">Выгрузить в Excel</button>
 </form>
 <div class="results hidden" id="results">
 <h2>Результаты:</h2>
 <table id="resultsTable">
 <thead>
 <tr>
 <th>Тип</th>
 <th>Сумма</th>
 </tr>
 </thead>
 <tbody>
 <!-- Результаты добавятся здесь -->
 </tbody>
 </table>
 <p><strong>Общая сумма:</strong> <span id="totalSum">0</span></p>
 </div>
 </div>
 <script>
 // Показ результатов
 function showResults(data) {
 const resultsDiv = document.getElementById('results');
 const downloadBtn = document.getElementById('downloadBtn');
 const resultsTable = document.getElementById('resultsTable').getElementsByTagName('tbody')[0];
 const totalSumSpan = document.getElementById('totalSum');

 // Очищаем таблицу
 resultsTable.innerHTML = '';

 // Заполняем таблицу данными
 let totalSum = 0;
 data.results.forEach(item => {
 const row = resultsTable.insertRow();
 const cell1 = row.insertCell(0);
 const cell2 = row.insertCell(1);
 cell1.textContent = item.type;
 cell2.textContent = item.sum.toFixed(2);
 totalSum += item.sum;
 });

 // Обновляем общую сумму
 totalSumSpan.textContent = totalSum.toFixed(2);

 // Показываем результаты
 resultsDiv.classList.remove('hidden');

 // Показываем кнопку "Выгрузить в Excel"
 downloadBtn.classList.remove('hidden');
 }

 // Скачать Excel
 function downloadExcel() {
 window.location.href = '/download';
 }

 // Отправка формы
 document.getElementById('uploadForm').addEventListener('submit', async function (e) {
 e.preventDefault();

 const formData = new FormData(this);

 try {
 const response = await fetch('/upload', {
 method: 'POST',
 body: formData
 });

 if (response.ok) {
 const result = await response.json();
 if (result.error) {
 alert(result.error);
 } else {
 showResults(result);
 }
 } else {
 const errorData = await response.json();
 alert(errorData.error || 'Ошибка при расчете');
 }
 } catch (error) {
 alert('Ошибка: ' + error.message);
 }
 });
 </script>
</body>
</html>

body {
 font-family: 'Arial', sans-serif;
 background-color: #f4f4f9;
 margin: 0;
 padding: 0;
 display: flex;
 justify-content: center;
 align-items: center;
 min-height: 100vh;
 background-image: linear-gradient(135deg, #e0eafc, #cfdef3);
}

/\* Контейнер формы \*/
.form-container {
 background: white;
 padding: 2rem;
 border-radius: 12px;
 box-shadow: 0 8px 16px rgba(0, 0, 0, 0.1);
 max-width: 450px;
 width: 100%;
 animation: fadeIn 0.6s ease-out;
}

/\* Заголовок \*/
h1 {
 font-size: 1.8rem;
 color: #333;
 margin-bottom: 1.5rem;
 text-align: center;
}

/\* Поля формы \*/
label {
 font-size: 1rem;
 margin-bottom: 0.5rem;
 display: block;
 color: #555;
}

input[type="file"],
input[type="number"],
select {
 width: 100%;
 padding: 0.75rem;
 margin-bottom: 1rem;
 border: 1px solid #ddd;
 border-radius: 8px;
 font-size: 1rem;
 transition: border-color 0.3s ease;
}

input[type="file"]:focus,
input[type="number"]:focus,
select:focus {
 border-color: #007bff;
 outline: none;
}

/\* Кнопки \*/
button {
 width: 100%;
 padding: 0.75rem;
 background-color: #007bff;
 color: white;
 border: none;
 border-radius: 8px;
 font-size: 1rem;
 cursor: pointer;
 transition: background-color 0.3s ease, transform 0.3s ease;
}

button:hover {
 background-color: #0056b3;
 transform: translateY(-3px);
 box-shadow: 0 6px 12px rgba(0, 0, 0, 0.2);
}

.hidden {
 display: none !important; /\* Добавляем !important для приоритета \*/
}

/\* Таблица результатов \*/
.results {
 margin-top: 2rem;
 padding: 2rem;
 background-color: white;
 border-radius: 12px;
 box-shadow: 0 8px 16px rgba(0, 0, 0, 0.1);
 animation: fadeIn 0.6s ease-out;
}

table {
 width: 100%;
 border-collapse: collapse;
 margin-top: 1rem;
}

table, th, td {
 border: 1px solid #ddd;
}

th, td {
 padding: 12px;
 text-align: left;
}

th {
 background-color: #f8f9fa;
 font-weight: bold;
}

tr:nth-child(even) {
 background-color: #f9f9f9;
}

tr:hover {
 background-color: #f1f1f1;
}

#totalSum {
 font-weight: bold;
 color: #28a745;
}

/\* Анимации \*/
@keyframes fadeIn {
 from {
 opacity: 0;
 transform: translateY(10px);
 }
 to {
 opacity: 1;
 transform: translateY(0);
 }
}

#uploadForm button {
 margin-bottom: 10px; /\* Отступ снизу \*/
 display: block; /\* Размещает кнопки друг под другом \*/
 width: 100%; /\* Чтобы они были одинаковой ширины \*/
}

#downloadBtn {
 display: none; /\* Скрыть кнопку по умолчанию \*/
}