~\ES\app5.py

```python
import streamlit as st
import pandas as pd
import numpy as np
from typing import Union

def calculate_scene_metrics(scene_df):
    scene_df['CV'] = scene_df[['Wpolygons', 'Wvertex', 'Wobject', 'Wlight', 'Wmaterials']]
.sum(axis=1) / 5
    total_frames = scene_df['frames'].sum()
    scene_df['FV'] = scene_df['frames'] / total_frames
    scene_df['ASS'] = scene_df['CV'] + scene_df['FV']
    scene_df['scene_speed_rank'] = abs(scene_df['ASS'] * 10).astype(int)
    return scene_df

def calculate_scene_speed_total_rank(scene_df: pd.DataFrame) -> int:
    scene_speed_total_rank = scene_df['scene_speed_rank'].mean()
    scene_speed_total_rank = max(1, min(10, scene_speed_total_rank))
    return int(round(scene_speed_total_rank))

def find_matching_gpu(gpu_df, rank_value, rank_type):
    matching_gpu = gpu_df[gpu_df[rank_type] == rank_value]
    if not matching_gpu.empty:
        return matching_gpu.iloc[0]['GPU_model']
    else:
        return "No matching GPU found"

def calculate_score(value):
    score = 10 * abs(value) / 30
    return max(1, min(10, score))

def determine_rank_type_and_value(option, scene_speed_total_rank, aus, SP, C, E):
    if option == 'System Analysis & User Preference':
        if scene_speed_total_rank > 8:
            rank_type = 'speed_rank'
            rank_value = max(scene_speed_total_rank, SP)
        elif aus < 5:
            rank_type = 'speed_rank'
            rank_value = scene_speed_total_rank
        else:
            rank_values = {'speed_rank': SP, 'cost_rank': C, 'energy_rank': E}
            rank_type = max(rank_values, key=rank_values.get)
            rank_value = rank_values[rank_type]
    elif option in ['Ultimate Speed', 'Ultimate Cost Saving', 'Ultimate Energy Saving']:
        rank_type = option.lower().replace("ultimate ", "").replace(" saving", "") + '_rank'
        rank_value = 10
    else:
        raise ValueError("Invalid option selected")

    return rank_type, rank_value


def main():
    st.set_page_config(layout="wide")

    st.title("++User preferences & Render Farm Management++")

# Initialize variables
    SP = C = E = 5  # Default values for sliders
    # First row
    st.subheader("Knowledge Base")
    col1, col2, col3 = st.columns(3)

    with col1:
        st.subheader("Upload Scene and GPU Files")
        scene_file = st.file_uploader("Choose a scene file", type=["xlsx"], key="scene_file")
        gpu_file = st.file_uploader("Choose a GPU file", type=["xlsx"], key="gpu_file")

    with col2:
        # User Preferences
        st.subheader("User Preferences")
        SP = st.slider("Select Speed", 1, 10, 5)
        C = st.slider("Select Cost", 1, 10, 5)
        E = st.slider("Select Energy", 1, 10, 5)

        SP_score = calculate_score(SP)
        C_score = calculate_score(C)
        E_score = calculate_score(E)
        aus = SP_score + C_score + E_score



    with col3:
        # Ultimate Selection
        st.subheader("Ultimate Selection")
        option = st.radio(
            "Select GPU Selection Strategy",
            ('System Analysis & User Preference', 'Ultimate Speed', 'Ultimate Cost Saving', '
Ultimate Energy Saving')
        )

    # Processed Scene Data
    st.subheader("Processed Scene Data")
    scene_data = None

    if scene_file:
        scene_df = pd.read_excel(scene_file)
        updated_scene_df = calculate_scene_metrics(scene_df)
        st.write(updated_scene_df)
        scene_data = updated_scene_df

    # Matching GPU
    st.subheader("Matching GPU")
    matching_gpu_model = None

    if scene_data is not None and gpu_file is not None:
        scene_speed_total_rank = calculate_scene_speed_total_rank(scene_data)
        rank_type, rank_value = determine_rank_type_and_value(option, scene_speed_total_rank,
aus, SP, C, E)

        st.write("Performance Speed Score: ", SP_score)
        st.write("Cost Score: ", C_score)
        st.write("Energy Score: ", E_score)
        st.write("+++Aggregated User Score (AUS): ", aus)
        st.write("Scene Speed Total Rank:", scene_speed_total_rank)
        st.write("Rank Type:", rank_type)
        st.write("Rank Value:", rank_value)

        gpu_df = pd.read_excel(gpu_file)
        matching_gpu_model = find_matching_gpu(gpu_df, rank_value, rank_type)
        st.write("Matching GPU Model: ", matching_gpu_model)

if __name__ == "__main__":
    main()

```