Jesteś starszym programistą Python i architektem aplikacji desktop ERP (CustomTkinter) z backendem Supabase (Postgres + Storage). Masz wykonać wdrożenie produkcyjne modułu **Nesting** na bazie gotowego, przetestowanego skryptu test_nesting_integration.py.

Kontekst

- Projekt to ERP dla cięcia laserowego.
- GUI: CustomTkinter (główne okno aplikacji).
- Backend: Supabase Postgres.
- Moduły obecne: produkty, oferty (quotations), zamówienia (orders), pozycje (order_items, quotation_items), materiały (materials_dict), cenniki/koszty (w istniejących plikach projektu).
- Masz dostęp do repo/ZIP z aktualnym rozwiązaniem oraz pliku test_nesting_integration.py (działa lokalnie).
- Docelowo moduł Nesting ma być zintegrowany z ofertami i zamówieniami: **po wczytaniu listy detali do oferty/zamówienia** użytkownik może przeprowadzić nesting/pakietowanie na arkuszach i uzyskać kosztorys.

Cel (MVP)

1. Dodać w GUI **oddzielną opcję/zakładkę/menu: "Nesting"** w głównym oknie programu, zintegrować z istniejącą nawigacją.
2. Zintegrować test_nesting_integration.py jako silnik nestingu:
    - wejście: lista detali (DXF/2D) z oferty/zamówienia,
    - wyjście: wynik nestingu (arkusze, rozmieszczenie, utilization, statystyki, geometria/parametry do kosztów).
3. Rozbudować moduł wyceny: policzyć koszty materiału/cięcia/operacyjne i koszty "na zlecenie" + koszty technologii, opakowań, transportu.
4. Przebudować GUI pod nowe pola i przepływ pracy.
5. Okno "Nesting" ma domyślnie zajmować **100% ekranu**.

Wymagania funkcjonalne (szczegółowo)

A) Integracja w GUI

- Dodaj w głównym oknie programu nową pozycję: **Nesting** gdzie użytkonik będzie mógł zwizualizować zlecenie – pytanie ofertowe i wysłać wyniki do modułu Zamówienia / oferty
- W module Nesting użytkownik ma móc:
    1. Wybrać kontekst: **Oferta** albo **Zamówienie** lub tylko **Test** (wybór konkretnego quotation_id lub order_id lub stworzenie nowych).
    2. Załadować listę detali/pozycji (z plików źródłowych DXF/DWG).
    3. Uruchomić nesting (na bazie test_nesting_integration.py).
    4. Zobaczyć wynik: lista arkuszy + utilization + koszty na arkusz i na zlecenie.
    5. Przesłać wyniki nestingu oraz parametry kosztów (checkboxy).
- W module Zamówienia i Ofert użytkownik będzie mógł wywołać opcję nesting i ma móc:
    1. Załadować listę detali/pozycji (z plików źródłowych DXF/DWG).
    2. Uruchomić nesting (na bazie test_nesting_integration.py).
    3. Zobaczyć wynik: lista arkuszy + utilization + koszty na arkusz i na zlecenie.
    4. Wybrać parametry kosztów (checkboxy + pola kwotowe), przeliczyć koszty jednym kliknięciem.
    5. Przesłać wyniki nestingu oraz parametry kosztów (checkboxy). Do GUI Zamówienia / Ofert
    6. Zapisać gotową wycenę do oferty/zamówienia (wartości i breakdown kosztów w DB).
    7. Wygenerować komplet dokumentów

GUI Nesting ma być niemal identyczne z obecny z pliku test_nesting_integration.py. ale zawierać opcję wysłania wyników do istniejącego bądź nowego zamówienia / oferty. (jeśli został uruchomiony z istniejącego zamówienia / oferty mamy to na sztywno) oraz klawisz Prześlij i Anuluj aktywne w zależności od kontekstu

GUI Zamówienia / **Oferty** ma zawierać:

- lista detali (nazwa, materiał, grubość, qty, thumbnails),
- panel parametrów kosztowych (checkboxy + stawki),

- panel wyników (arkusze i koszty),
- przyciski: Run Nesting / Recalculate / Save.
- Nesting ma działać w formie dwustronnego przepływu – jeśli na liście detali zamówienia / oferty są już załadowane pozycje, mają być wysłane jako dane wejściowe do nesting (lokalizacje plików), jeśli w panelu Nestingu wybrano Prześlij lista detali musi być zaktualizowana, wygenerowane nowe thumbnails.
- Oczekiwany wygląd modułów przesyłam w załączonej grafice
- Po zatwierdzeniu zamówienia / oferty należy zaktualizować dane w supabase wraz z aktualizacją/dodaniem produktów, załączników itd

B) Koszty — wymagane elementy

Masz obsłużyć i dodać do bazy pola/konfiguracje kosztów:

1. **Koszt materiału** – liczony z arkuszy po nestingu.
2. **Koszt cięcia** – z cennika (m/min) + oszacowanie czasu.
3. **Koszt usuwania folii**:
    - checkbox w GUI,
    - domyślnie **TRUE dla blach nierdzewnych do 5mm grubości**,
    - domyślnie FALSE dla pozostałych,
    - dodać pole kosztowe w Supabase (konfigurowalne stawki w zależności od materiału; jeśli wolisz: osobna tabela konfiguracyjna).
    - prędkość zdejmowania folii: **średnio 15 m/min**. Dodaj pole koszty godzinowego do wypełnienia przez operatora
4. **Koszt przebicia / wpaleń (piercing)**:
    - checkbox w GUI,
    - dodać pole kosztowe w Supabase **oddzielnie dla każdej grubości i gatunku blachy** (tabela stawek),
    - czas wpalenia szacuj:
        - inox: **0.5–1.0 s** (rośnie z grubością),
        - czarna: **0.5–2.0 s** (rośnie z grubością),
    - liczbę przebić oszacuj z geometrii (otwory/contours) albo heurystyką (jeśli brak).
5. **Koszty technologii “na zlecenie”** – pole kwotowe per order/quotation.

6. **Koszty operacyjne per arkusz**:
    - domyślnie **40 PLN** za załadunek+rozładunek arkusza,
    - liczone *per każdy użyty arkusz* w wyniku nestingu.
7. **Koszt opakowań na zlecenie** – pole kwotowe.
8. **Koszty transportu na zlecenie** – pole kwotowe.
9. Dodaj **+25% zapasu** na czas/niepewność do kalkulacji całkowitego czasu procesu (cięcie + piercing + folia).
10. Przy wycenie zaproponuj **dwa warianty**:

- **Wariant A:** koszt "cennikowy" (prosty, deterministyczny) – na bazie długości cięcia, prędkości z cennika, stawki pierce, folii, operacyjnych.
- **Wariant B:** koszt "czasowy" (rozszerzony) – z oszacowaniem czasu na podstawie:
    - długość ścieżki cięcia,
    - liczba wpaleń,
    - "skomplikowanie" nestingu/cutting path,
    - prędkości m/min z cennika,
    - dodatkowe czasy (folia),
    - +25% bufora.
- Wariant B może używać heurystyk/ML, ale **MVP ma działać bez ML** (ML może być opcjonalny/future flag).

C) Obliczanie rzeczywistego kosztu materiału — poprawa metody

Masz przeanalizować obecną metodę (którą stosowałem):

- Zmniejsz wysokość arkusza do rzeczywistego wykorzystania w osi Y
- Zmniejszony arkusz traktuję jako 100% materiału,
- jeśli utilization arkusza to X%, to każdy detal ma wagę/koszt dzielony przez X (np. 10 kg przy 80% => 12.5 kg do kalkulacji).

Twoje zadanie:

1. **zrobić research i zaproponować lepszy model** alokacji kosztu materiału (najlepiej praktyczny, stosowany w branży),
2. opisać go jasno i wdrożyć w kodzie z możliwością przełączenia (np. dropdown: "Allocation model"),

3. zachować wsteczny model jako opcję “Legacy”.

Wymóg dodatkowy:

- **Po nestingu zmniejsz obszar arkusza w osi Y do rzeczywistego zajętego rozmiaru**, **zostaw szerokość w osi X** (bounding box po ułożeniu) i dopiero to traktuj jako “zużyty arkusz” do kosztu (zamiast pełnego nominalnego arkusza), o ile to nie psuje realiów technologicznych. Jeżeli to niepoprawne technologicznie, zaproponuj alternatywę (np. „remnant management”, minimalna długość arkusza, itd.) i uzasadnij.

Ważne: Research ma być wykonany z podaniem źródeł (linki/cytaty do praktyk branżowych / nesting costing / sheet utilization allocation). Jeśli środowisko agenta pozwala: przeszukaj web. Jeśli nie: opisz założenia i co byś sprawdził.