так, погоджуюсь. Твій оновлений **КРОК 2** повністю відповідає нашим цілям — сумісність із наявною панеллю, продуктивність і детермінізм бекенду, точність правил і **інновації** (ordinal-ризик, zero-copy spans, idempotency). Нижче — компактний, “бойовий” план у форматі **Architecture/Design → Plan → Code → Tests → Validation → Next steps** з чіткими вимогами до коду.

**Architecture / Design (що і навіщо)**

* **Комбо-відповідь** /api/analyze: повертаємо **одночасно** legacy (analysis/results/clauses) для поточної панелі й **SSOT** (document) для нових можливостей.
* **Енвелоп** кожної відповіді: schema\_version, response\_version, cid (дзеркало x-cid), generated\_at — полегшує трасування/сумісність.
* **Zero-copy spans**: усі Span{start,length} прив’язані до **оригінального буфера**. Додаємо **anchors**: pre\_hash/post\_hash для переприв’язки після tracked changes.
* **Ordinal ризику** (low=0, medium=1, high=2, critical=3): єдина шкала для агрегування/дельт у /api/qa-recheck.
* **Idempotency** для /api/analyze: x-idempotency-key (або sha256(normalized\_text|policy\_pack)); **LRU(128 / 10min)** in-memory.
* **Надійність/швидкість**: CORS лише https://localhost:3000, семафор 4, таймаути 15s/5s, body-limit 2.5MB, Problem+JSON на всі помилки.
* **Explainability**: trace[], timings{identify\_ms,rules\_ms,cross\_ms,total\_ms}, cross\_refs (мінімальний CKG).

**Plan (покроково, по файлах з “Done when”)**

**1) core/schemas.py**

**Що додати**

* Константи та хелпери ризику:
  + RISK\_ORDER = {"low":0,"medium":1,"high":2,"critical":3}
  + risk\_to\_ord(str)->int / ord\_to\_risk(int)->str (клемп 0..3)
* Коментар до DeltaMetrics.risk\_delta (використовує ординал).
* (Якщо ще не додано) DTO QARecheckIn/Out.

**Done when:** імпорти з інших модулів працюють; QA-recheck повертає дельти саме за цією шкалою.

**2) contract\_review\_app/api/app.py**

**Що додати/закріпити**

* **CORS**: https://localhost:3000 (+127.0.0.1).
* **Middleware**:
  + x-cid (UUIDv4) → у response.headers["x-cid"] і в енвелоп.
  + **Body-limit 2.5MB** (413 + Problem+JSON).
  + **Semaphore(4)** + asyncio.wait\_for (15s analyze / 5s draft).
  + Уніфікований **Problem+JSON** error-handler.
* **Idempotency LRU** для /api/analyze:
  + key = headers["x-idempotency-key"] || sha256(normalized\_text|policy\_pack).
  + при хіті — повертати кешований JSON (додаючи свіжий cid).
* **Ендпоїнти**:
  + GET /health → {status, schema\_version, rules\_count, uptime\_ms}.
  + POST /api/analyze → **AnalyzeOut** (legacy + SSOT) **в енвелопі**.
  + POST /api/gpt/draft → DraftOut (rule-based fallback guaranteed).
  + POST /api/suggest\_edits → SuggestOut (≥1, top-k ≤10).
  + POST /api/qa-recheck → QARecheckOut (дельти через ординал).

**Done when:** повторний /api/analyze з тим самим ключем/текстом → відповідає з кеша; усі відповіді містять cid/версії/час.

**3) contract\_review\_app/api/orchestrator.py**

**Що реалізувати**

* run\_analyze(inp: AnalyzeIn) -> AnalyzeOut
  1. doc = pipeline.analyze\_document(...) → заповнені index/analyses/cross\_refs/timings.
  2. (analysis, results, clauses) = pipeline\_compat.to\_panel\_shape(doc).
  3. Зібрати **енвелоп** + AnalyzeOut.
* run\_draft(inp: DraftIn) -> DraftOut
  1. якщо analysis.proposed\_text є — повернути його; інакше pipeline.synthesize\_draft(analysis\_or\_text, mode).
* run\_suggest(inp: SuggestIn) -> SuggestOut
  1. нормалізувати картки {title, draft, reason, sources[], span?, clause\_id, risk}.
* run\_qa\_recheck(inp: QARecheckIn) -> QARecheckOut
  1. prev = analyze\_document(text\_before) → агрегуємо ризик як **max ord**;
  2. застосувати applied\_changes до тексту (простий текстовий патч по spans);
  3. now = analyze\_document(text\_after) → порахувати risk\_delta = ord(now)-ord(prev), status\_from/to, score\_delta, residual\_risks(top-3).

**Done when:** функції **ніколи** не віддають сирі ексепшени назовні; JSON відповідає схемам.

**4) contract\_review\_app/engine/intake.py**

**Що додати**

* segment\_document(text, language=None) -> DocIndex
  + Детерміновані евристики (headers/lists/“Section/Clause/Article”).
  + **Zero-copy spans**: start/length у координатах вхідного text.
  + Clause.id стабільний: siphash(start|length|sha256(text[start:start+length])).
  + pre\_hash/post\_hash: sha256(контексту 32–64 символи до/після).

**Done when:** однаковий уривок ⇒ ті ж Span/id; порожній текст ⇒ валідний DocIndex(clauses=[]).

**5) contract\_review\_app/legal\_rules/registry.py + правила**

**Контракт правила**

def rule\_<topic>(text: str, ctx: dict) -> tuple[list[Finding], Optional[str], list[Citation]]:

...

* ctx: clause\_type, сусідні тексти, jurisdiction, policy\_pack.
* Пер-rule timeout (10–20 ms) та “circuit breaker” для важких regex (у виклику).

**Done when:** зміна ключової умови в тексті → очікуваний Finding змінюється; правила не блокують двигун.

**6) contract\_review\_app/engine/executor.py**

**Що зробити**

* run\_rules(index, policy\_pack=None) -> list[AnalysisOutput]
  + рахує score/risk/severity/status (інваріанти зберігаються), додає proposed\_text/citations/trace.
* compute\_cross\_refs(analyses) -> list[CrossRef]
  + мінімум: (Termination↔LoL), (Jurisdiction↔GoverningLaw), (NDA↔DataProtection), (ForceMajeure↔SLA/LD).

**Done when:** аналіз різних уривків → різні findings; cross-refs побудовані.

**7) contract\_review\_app/engine/pipeline.py**

**Що закріпити**

* analyze\_document(...) -> DocumentAnalysis
  + повертає index, analyses, cross\_refs, overview, timings{identify\_ms,rules\_ms,cross\_ms,total\_ms}.
* synthesize\_draft(analysis|dict, mode="friendly") -> str
  + 3 стилі; якщо analysis.proposed\_text уже є — повертаємо його.

**Done when:** на одному уривку три стилі дають **відмінні** тексти; timings заповнені.

**8) contract\_review\_app/engine/pipeline\_compat.py**

**Що зробити**

* to\_panel\_shape(doc: DocumentAnalysis) -> (analysis, results, clauses)
  + clamp полів (довжини, кількість елементів), safe-defaults.

**Done when:** /api/analyze на порожньому тексті повертає валідний JSON; панель не ламається.

**Code (мінімальні вставки — “якірні” фрагменти)**

**1) Risk ordinal (schemas.py)**

RISK\_ORDER = {"low":0,"medium":1,"high":2,"critical":3}

def risk\_to\_ord(r: str) -> int:

return max(0, min(3, RISK\_ORDER.get((r or "medium").lower(), 1)))

def ord\_to\_risk(n: int) -> str:

keys = ["low","medium","high","critical"]

return keys[max(0, min(3, int(n)))]

**2) Idempotency LRU (app.py)**

# pseudo

IDEMP\_CACHE = LruDict(max\_items=128, ttl\_seconds=600)

def make\_idem\_key(req\_body: dict, headers: dict) -> str:

k = headers.get("x-idempotency-key")

if k: return k

# normalize policy + text

import hashlib, json

payload = json.dumps({"text": req\_body.get("text",""), "policy": req\_body.get("policy\_pack")}, sort\_keys=True)

return hashlib.sha256(payload.encode("utf-8")).hexdigest()

**3) QA-recheck дельти (orchestrator.py)**

prev\_ord = max(risk\_to\_ord(a.risk or "medium") for a in prev.analyses) if prev.analyses else 0

now\_ord = max(risk\_to\_ord(a.risk or "medium") for a in now.analyses) if now.analyses else 0

risk\_delta = now\_ord - prev\_ord # -3..+3

status\_from = prev.overview.status

status\_to = now.overview.status

**4) Anchors для spans (intake.py)**

def \_anchor\_hash(s: str) -> str:

import hashlib

return hashlib.sha256(s.encode("utf-8")).hexdigest()[:16]

pre = \_anchor\_hash(text[max(0,start-64):start])

post = \_anchor\_hash(text[start+length:start+length+64])

**Tests (unit / property / e2e)**

**Unit**

* risk\_to\_ord/ord\_to\_risk: межі 0..3; невідоме значення → 1 (“medium”).
* Idempotency: однаковий make\_idem\_key для семантично однакового body.
* Compat: рядок у citations перетворюється на Citation[].
* Cross-refs: для контрприкладів (невідповідна пара) ребро не створюється.

**Property**

* segment\_document() завжди дає неперекривні спани в межах тексту.
* analyze\_document() не падає на довільному тексті; score ∈ [0..100].

**e2e (smoke)**

* /health=200; /api/analyze → комбо-відповідь (legacy+SSOT з енвелопом і cid).
* Повторний /api/analyze з тим самим key/text → **швидше**, кеш-хіт.
* /api/gpt/draft → model="rule-based" і draft\_text ≥ 20.
* /api/qa-recheck на “після правок” → risk\_delta ≤ 0, status\_from/to узгоджені.

**Validation (критерії приймання)**

* Жодних 5xx на валідних payload; **Problem+JSON** на помилки/таймаути/413.
* p95: /api/analyze ≤ 1200 мс (повтор — <200 мс через кеш), /api/gpt/draft ≤ 500 мс.
* Панель працює без змін; бейджі/списки відображаються як зараз.
* timings, trace, cross\_refs, anchors присутні; Span абсолютний (zero-copy).

**Next steps**

1. Внести risk-хелпери / (якщо треба) QARecheck\* у schemas.py.
2. Доробити app.py (LRU idempotency, семафор, таймаути, Problem+JSON, /api/qa-recheck, envelope).
3. Завершити orchestrator.py (всі 4 функції).
4. Стабілізувати intake.py (anchors) та pipeline.py (timings).
5. Прогнати тести (unit/property/e2e smoke).
6. Перейти до **КРОКУ 3**: анотації у Word (коментарі на проблемах за Span, jumpToSpan) і UI-дельти з /api/qa-recheck.

**STATE LOG**

**Ціль:** КРОК 2 = прод-готові API з комбо-відповідями, idempotency, ordinal-дельтами, zero-copy spans, envelope.  
**Артефакти:** специфікації на зміни у schemas/app/orchestrator/intake/executor/pipeline/compat.  
**TODO:** реалізувати пункти 1→5, перевірити панель; підготуватися до КРОКУ 3 (анотації та UI-дельти).  
**Ризики:** важкі regex у правилах (мітити timeout/circuit-breaker), payload>2.5MB (413), кеш WEF (вирішено версіонуванням і скриптами).