так, я повністю погоджуюсь із планом. Перший артефакт — саме core/schemas.py. Нижче даю **чітку специфікацію** цього файла: які моделі мають бути, які поля, типи, інваріанти, що вважається “готово”, і як ми далі цим користуємось. Це — “SSOT-контракт” між усіма модулями і панеллю Word.

**Що має бути в core/schemas.py (канон)**

**0) Загальні вимоги**

* **Єдине джерело істини (SSOT)** для всіх API/engine/UI.
* Pydantic v2 (або сумісність із v1 через мінімальні шими).
* **Сувора валідація** + **толерантні нормалізатори** для зворотної сумісності (списки рядків/об’єктів; порожній вхід → валідні дефолти).
* Поле schema\_version: str = "1.0" у документ-рівневих відповідях.
* **Діапазони/інваріанти** (див. нижче) перевіряються валідаторами.

**1) Переліки (enums)**

* Status = "OK" | "WARN" | "FAIL"
* RiskLevel = "low" | "medium" | "high" | "critical"
* Severity = "info" | "minor" | "major" | "critical"
* DraftMode = "friendly" | "standard" | "strict"

Інваріанти:  
Severity="critical" ⇒ RiskLevel∈{"high","critical"} ⇒ Status="FAIL"  
Якщо findings=[] ⇒ Status="OK" та RiskLevel="low" (як дефолт).

**2) Примітиви/Value-objects**

* **Span**: { start: int, length: int, page: int|None, block: int|None }  
  Інваріанти: start ≥ 0, length ≥ 0.
* **Citation**: { system: "UK"|"UA"|"EU"|"INT", instrument: str, section: str, url: str|None }
* **CrossRef**: { source\_clause\_id: str, target\_clause\_id: str, relation: "contradicts"|"depends\_on"|"duplicates"|"requires" }

**3) Блоки знаходок та порад**

* **Finding**  
  Поля:  
  code: str (стабільний ідентифікатор правила),  
  message: str (людинозрозуміле),  
  severity: Severity,  
  risk: RiskLevel|None (якщо не задано — виводиться з severity),  
  evidence: str|None (цитата з тексту),  
  span: Span|None,  
  citations: list[Citation] = [],  
  tags: list[str] = [].

Нормалізатори: приймає citations як рядок/список рядків і перетворює у структурні Citation.

* **Recommendation**  
  Поля:  
  code: str|None,  
  text: str,  
  rationale: str|None,  
  citations: list[Citation] = [].
* **Diagnostic** (для трейсингу/логів engine, не для користувача)  
  rule: str, message: str, severity: "info"|"warn"|"error", legal\_basis: list[str] = [].
* **Suggestion** (для блоку *Suggested edits* у панелі)  
  text: str (alias message), reason: str|None, span: Span|None, source: str|None ("rule"/"gpt").

**4) Рівень клауза (unit)**

* **Clause**  
  id: str, type: str, title: str|None, text: str, span: Span.
* **AnalysisOutput** (результат по одній клаузі з rule-двигуна)  
  Обов’язкові:  
  clause\_id: str, clause\_type: str, text: str, status: Status.  
  Оцінки:  
  score: int = 0 (0..100), risk: RiskLevel = "medium", severity: Severity = "minor".  
  Вміст:  
  findings: list[Finding] = [],  
  recommendations: list[Recommendation] = [],  
  proposed\_text: str = "" (якщо правило сформувало зразок),  
  suggestions: list[Suggestion] = [],  
  citations: list[Citation] = [],  
  diagnostics: list[Diagnostic] = [],  
  trace: list[str] = [],  
  cross\_refs: list[CrossRef] = [],  
  metadata: dict[str,str] = {}.

Інваріанти: 0 ≤ score ≤ 100. Якщо хоч один Finding.severity="critical" ⇒ status="FAIL".

**5) Рівень документу**

* **DocIndex**  
  document\_name: str|None, language: str|None, clauses: list[Clause].
* **DocumentAnalysis**  
  schema\_version: str = "1.0",  
  document\_name: str|None,  
  summary\_score: int (0..100),  
  summary\_risk: RiskLevel,  
  summary\_status: Status,  
  residual\_risks: list[Finding] = [] (некритичні, що лишились),  
  analyses: list[AnalysisOutput],  
  cross\_refs: list[CrossRef] = [],  
  index: DocIndex.

Інваріанти: агрегати консистентні зі складовими (наприклад, якщо є FAIL серед analyses ⇒ summary\_status=FAIL).

**6) DTO для публічних API**

**Примітка:** Панель Word зараз очікує “спрощений” analysis із полями (score/risk/status/severity/findings/recommendations/proposed\_text). Тому в схемах мають бути **і SSOT-типи, і “панельні” DTO** (compat).

* **Analyze**
  + AnalyzeIn — { document\_name?: str, text: str, language?: str }
  + AnalyzeOut — **дві частини**:
    1. **Panel-shape** (для сумісності зі сьогоднішнім UI):  
       analysis: { clause\_type: str, findings: list[Finding], recommendations: list[str], proposed\_text: str, score: int, risk: str, severity: str, status: Status },  
       results: dict (сирі груповані дані для панелі),  
       clauses: list[dict] (спрощені клауза з offset).
    2. **SSOT-shape** (нове, сучасне):  
       document: DocumentAnalysis.
  + Вимога: **завжди** повертати обидва блоки (панельний — для старого рендеру; SSOT — для нової анотації/QA).
* **GPT Draft**
  + DraftIn — { text?: str, analysis?: AnalysisOutput|dict, clause\_type?: str, mode?: DraftMode="friendly" }
  + DraftOut — { draft\_text: str, alternatives?: list[str], rationale?: str, citations\_hint?: list[Citation], model?: str="rule-based" }
* **Suggest**
  + SuggestIn — { clause\_id?: str, clause\_type: str, text: str, mode: DraftMode }
  + SuggestOut — { suggestions: list[Suggestion] }
* **QA-Recheck**
  + QARecheckIn — { document\_name?: str, text: str, applied\_changes?: list[{clause\_id: str, before: str, after: str}] }
  + QARecheckOut — { deltas: { score\_delta: int, risk\_delta: int, status\_from: Status, status\_to: Status }, residual\_risks: list[Finding] }

**7) Нормалізатори/сумісність (must-have)**

* Приймаємо у вхідних/внутрішніх моделях:
  + citations як рядок/список рядків → перетворюємо у Citation.
  + suggestions як рядки або {message: "...", reason?: "..."} → у Suggestion(text=...).
  + Порожні або відсутні поля → безпечні дефолти (щоб **ніколи** не ламати панель).
* Усі моделі мають model\_config = ConfigDict(populate\_by\_name=True, extra="ignore").

**8) Інваріанти/перевірки (валидація)**

* score ∈ [0..100]; summary\_score — середнє або вагове середнє analyses[].score.
* Якщо findings=[] і proposed\_text=="" → status не може бути FAIL.
* Якщо є Finding.severity="critical" → status="FAIL".
* Span.start ≥ 0, Span.length ≥ 0.

**9) “Definition of Done” (для цього файла)**

1. У core/schemas.py **усі моделі** вище описані та протестовані валідаторами.
2. Файл експортує **рівно ці типи** (щоб інші модулі імпортували з одного місця).
3. Будь-який порожній текст, поданий у AnalyzeIn, дає валідний AnalyzeOut (обидві частини), без виключень.
4. JSON-приклади (для швидких ручних тестів) під рукою:
   * мінімальний AnalyzeOut (панель + SSOT),
   * DraftOut із alternatives,
   * SuggestOut із двома картками.

**Що у тебе вже є, і що додати**

У завантаженому schemas.py вже присутні: Finding, Diagnostic, Suggestion, AnalysisOutput (базовий), “panel-shape” Analysis/AnalyzeResponse.  
**Додати/уточнити:**

1. Enums RiskLevel, Severity, DraftMode.
2. Value-objects: Span, Citation, CrossRef.
3. Рівень клауза: Clause та обов’язкові поля clause\_id, risk, severity в AnalysisOutput (із валідними дефолтами).
4. Рівень документу: DocIndex, DocumentAnalysis.
5. DTO для API: AnalyzeIn/Out (із подвійною формою), DraftIn/Out, SuggestIn/Out, QARecheckIn/Out.
6. Нормалізатори для citations, suggestions (у тебе частина вже є — розширити до структурних Citation).
7. Інваріанти (валидація) і schema\_version.

**Як ми це використаємо далі**

* engine/intake.py формує DocIndex (клаузи+Span).
* engine/executor.py повертає list[AnalysisOutput] (по кожній клаузі).
* engine/pipeline\_compat.py робить **перетворення SSOT → panel-shape** для сьогоднішнього UI.
* api/app.py повертає **обидві** форми в AnalyzeOut, а також DraftOut/SuggestOut/QARecheckOut.
* word\_addin\_dev/taskpane.bundle.js може й надалі використовувати поточний рендер (panel-shape), а нові фічі (анотації за Span, повторний QA) візьмуть дані зі **SSOT-блоку**.

**Мінімальні приклади JSON для ручної перевірки (орієнтир)**

* **AnalyzeOut (урізано):**

{

"analysis": {

"clause\_type": "confidentiality",

"findings": [],

"recommendations": [],

"proposed\_text": "",

"score": 82,

"risk": "low",

"severity": "minor",

"status": "OK"

},

"results": {},

"clauses": [],

"document": {

"schema\_version": "1.0",

"document\_name": "contract.docx",

"summary\_score": 78,

"summary\_risk": "medium",

"summary\_status": "WARN",

"residual\_risks": [],

"analyses": [],

"cross\_refs": [],

"index": { "document\_name": "contract.docx", "language": "en", "clauses": [] }

}

}

* **DraftOut:**

{ "draft\_text": "Each party shall keep the Confidential Information strictly confidential...", "alternatives": [], "model": "rule-based" }

* **SuggestOut:**

{ "suggestions": [ { "text": "Add explicit right to seek injunctive relief.", "reason": "Remedy clarity", "source": "rule" } ] }

**Далі**

Якщо цей контракт тебе влаштовує — давай. Наступний крок після core/schemas.py — **engine/pipeline\_compat.py** (нормалізація до panel-shape та дефолти), а потім **api/app.py** (стабільні ендпоїнти, що повертають і panel-shape, і SSOT).