**Назва блоку:** citation\_resolver

**Завдання:**

1. **Pydantic-схеми I/O** – підготувати схеми введення/виведення для модуля цитування.
2. **Алгоритм matching** – реалізувати логіку знаходження відповідних юридичних норм для кожного finding.
3. **Метадані та трасування** – забезпечити додавання метаданих до цитат і прозорість (traceability) їх походження.
4. **Unit-тести** – розробити модульні тести для перевірки роботи citation\_resolver.
5. **Інтеграція з pipeline** – включити блок citation\_resolver у існуючий pipeline аналізу.
6. **Безпека та логування** – дотриматися вимог безпеки: відсутність утримання даних, аудит-логування, виключення PII.

**1. Pydantic-схеми I/O**

**Опис задачі:** Необхідно визначити формат даних, з яким працюватиме citation\_resolver. Вхідними даними є об’єкт Finding (визначений у core/schemas.py), а вихідними – об’єкти типів Citation та Evidence (і можливо нова структура для оцінки відповідності). Потрібно переконатися, що структура виходу відповідає Single Source of Truth схемам у core/schemas.py. Якщо в схемах бракує полів для цього блоку (наприклад, поля **score** або вкладеного **Evidence** в цитаті), слід додати їх у Pydantic-моделі. Зміни в схемах мають бути мінімальними і узгодженими з рештою системи. Всі модулі повинні використовувати ці оновлені моделі, тому після правок слід перевірити, що валідуються існуючі тести схем та сумісності.

**Очікуваний результат:** Оновлений файл core/schemas.py (diff), де додані або уточнені Pydantic-моделі для **Citation** (наприклад, додано поле score: float та поле для **Evidence** якщо його не було) і, за потреби, нова модель (наприклад, CitationResult або аналог) для результату роботи citation\_resolver. Код має пройти перевірки форматування (ruff/black), типів (mypy) і всі тести схем.

**Codex prompt:**

Modify the Pydantic models in `core/schemas.py` to support the citation\_resolver output.

1. Add any necessary fields to represent a legal citation’s metadata, associated evidence excerpt, and a confidence score. For example, ensure the Citation model can include an Evidence object (text snippet and source info) and a float score.

2. Do not remove or rename existing fields; use Optional[...] for new fields if appropriate to maintain backward compatibility.

3. Ensure the Finding model has a field for citations (e.g., `citation: Optional[Citation]` or `List[Citation]`) if not already present, so that the pipeline can attach the resolver’s output.

4. Keep changes minimal and consistent with the project style. All existing tests (schemas validation, etc.) must remain passing.

Provide the changes as a unified diff.

**2. Алгоритм matching**

**Опис задачі:** Потрібно реалізувати основний алгоритм, що бере Finding і повертає релевантну юридичну цитату. Алгоритм має **детерміновано** знаходити відповідні нормативні акти, правила або настанови, використовуючи задані джерела (legislation.gov.uk, керівництва FCA, ICO, стандарти OGUK тощо). Це означає, що на однаковий finding система завжди повертає ту саму цитату. На практиці реалізація може комбінувати пошук ключових слів та заздалегідь налаштовані відповідності:

* Розпізнати суть finding (яке правило спрацювало, яка тема). Наприклад, якщо finding стосується персональних даних – шукати в базі GDPR/закону про захист даних; якщо стосується фінансових послуг – звернутися до правил FCA; якщо контракт нафтовий – шукати стандарт OGUK тощо.
* Якщо для певного правила передбачена статична відповідність (mapping), використовувати її. (Наприклад, для правила відсутності положення про обробку даних – прямо повертати посилання на Article 28(3) UK GDPR без пошуку).
* Інакше виконати пошук у відповідному корпусі: запит до API Legislation.gov.uk або до внутрішнього індексу документів. Пошук має бути точним, без залучення LLM на цьому етапі (щоб забезпечити передбачуваність). За потреби – парсити результат (наприклад, витягнути текст розділу закону).

Алгоритм повинен зібрати дані цитати: назву джерела (закон, норматив або гайдлайн), точну адресу (наприклад, акт і розділ) та відповідний фрагмент тексту як **Evidence**. Результат супроводжується оцінкою **score** (наприклад, 1.0 для прямого співпадіння або менше, якщо знаходиться найбільш схожий текст). У випадку, якщо жодного джерела не знайдено, модуль має повертати None або порожню цитату, щоб pipeline знав, що посилання відсутнє.

*Причина:* Наразі поле citation у Findings майже не заповнюється, і користувач бачить лише рекомендацію без підтвердження. Це знижує довіру. Новий алгоритм закриє цю прогалину, автоматично підтягаючи релевантні зовнішні джерела (акти, стандарти) для обґрунтування кожного finding.

**Очікуваний результат:** Новий модуль (файл) core/citation\_resolver.py або відповідний клас/функція в існуючому просторі, що реалізує функцію resolve\_citation(finding: Finding) -> Optional[Citation] (чи List[Citation]). Код функції включає визначення логіки зіставлення: наприклад, використання словника відповідностей для відомих правил та виклик функції пошуку для загальніших випадків. Додатково, функції-хелпери: для пошуку у зовнішніх джерелах (можливо, через HTTP API чи локальний індекс). Очікується, що з’явиться новий код без зміни існуючих модулів (окрім імпорту нової функції), або мінімальні зміни у вже наявному шаблоні citation\_resolver. У diff повинен бути видно реалізацію алгоритму.

**Codex prompt:**

Create a new module `core/citation\_resolver.py` with a function `def resolve\_citation(finding: Finding) -> Optional[Citation]:` that returns a legal citation for a given finding.

Requirements:

- Use the definitions of Finding, Citation, and Evidence from core/schemas.py (already imported or import them).

- Implement a deterministic matching logic:

- If the Finding matches specific known cases (e.g., a rule about data protection or an Oil&Gas contract), return a predefined Citation with evidence.

- Otherwise, perform a keyword search in relevant sources. (You can simulate this with a stub function or predefined data, as external API calls are not performed in this context.)

- For example, if the finding indicates a missing personal data protection clause, return a Citation referencing “UK GDPR Article 28(3)” with an Evidence snippet explaining the requirement for such contracts.

- If the finding is related to an Oil & Gas industry contract (e.g., rule name contains "Oil&Gas"), return a Citation referencing OGUK guidelines (e.g., “OGUK Model Contract”) with a relevant snippet.

- Include a confidence score (e.g., 1.0 for a direct mapping or a value <1.0 for a best-effort match).

- If no relevant source is found, return None.

Ensure the function and any helper logic are clear, well-typed, and adhere to project style. Only modify/create the necessary files for this feature. Provide the implementation as a diff.

**3. Метадані та трасування**

**Опис задачі:** Для кожної знайденої цитати треба додати метадані, що підвищують її корисність і прозорість. Метадані включають: тип джерела та його реквізити (наприклад, “Legislation (UK Act)”, “FCA Guidance”, “ICO guideline”, “OGUK Standard” тощо), точну назву акта або документа, розділ/статтю. Ці поля можуть бути частиною моделі **Citation** (наприклад, source\_name, section\_title, section\_number). Крім того, треба зберегти трасування – тобто можливість відстежити, як і на основі чого було отримано цю цитату. У контексті нашого пайплайну трасування забезпечується передачею ідентифікатора сесії (cid) і позначенням правила, що згенерувало finding. Необхідно:

* Переконатися, що resolve\_citation при виконанні може отримати доступ до ідентифікатора запиту (через поле в Finding або глобальний контекст) і неявно чи явно використовувати його для логування.
* Додати до об’єкта Citation поля, які ідентифікують джерело (наприклад, URL або унікальний ID нормативного акта) для забезпечення traceability.
* Якщо у Finding є атрибут rule або source, передати/використати його для зв’язку Citation з першопричиною (наприклад, записати в лог: “Для правила X знайдено посилання Y”). Це гарантує можливість аудиту: можна буде простежити, яке правило породило яку цитату.

**Очікуваний результат:** Доповнення моделі **Citation** (у schemas.py) полями метаданих, якщо це не зроблено в попередньому кроці. Приклад: Citation.source: Literal["legislation", "FCA", "ICO", "OGUK", ...], Citation.link: HttpUrl (URL на джерело якщо доступно), Citation.title: str (назва акту чи документа), Citation.section: str (номер статті/секції або інше точне посилання всередині документа). У реалізації resolve\_citation – заповнення цих полів (наприклад, для GDPR Article 28(3) заповнити title="UK GDPR", section="Article 28(3)", source="legislation"), а також виклики логування з зазначенням cid та деталей (можливо, через logger.info). Очікується diff, що показує додані поля у схемах (якщо не були додані раніше) та оновлення функції, яка тепер встановлює метадані і виконує логування.

**Codex prompt:**

Enhance the citation output with metadata and ensure traceability.

Tasks:

1. Extend the Citation model (in core/schemas.py) with fields for source type, reference details, and link if applicable (only if not already present). For example: `source: str` (e.g., "legislation", "FCA", "ICO", "OGUK"), `link: Optional[AnyUrl]` for a URL, `title: str` for the document name, and `section: str` for section/article identifier.

2. Update the `resolve\_citation` function to populate these metadata fields for each returned Citation. E.g., if returning UK GDPR Article 28(3), set `citation.source="legislation"`, `citation.title="UK GDPR"`, `citation.section="Article 28(3)"`, and a relevant `link` (if available, e.g., an official URL).

3. Implement logging within `resolve\_citation` to record the operation. Use the existing logging utility (e.g., `logger` if configured) to log an info message containing the contract/request ID (cid) and a brief note of which finding (by rule name or id) is mapped to which citation reference. Do \*\*not\*\* log any sensitive text, only identifiers (cid, rule) and the citation reference.

4. Ensure these changes do not break any tests and follow coding standards.

Provide the unified diff for updated schemas and citation\_resolver implementation.

**4. Unit-тести**

**Опис задачі:** Розробити модульні тести, які перевіряють роботу citation\_resolver на різних сценаріях. Мета – впевнитися, що для заданих findings повертаються правильні цитати (або їх відсутність), що усі поля заповнені коректно, і що система поводиться детерміновано. Тести мають охоплювати:

* **Позитивні випадки**: коли для конкретного finding очікується відома юридична цитата. Наприклад, якщо finding стосується відсутності положення про обробку персональних даних, тест очікує посилання на відповідну норму (UK GDPR або DPA 2018) з правильним текстом evidence. Так само для галузевого кейсу: для правила, пов’язаного з нафто-газовим контрактом, очікується посилання на стандарт OGUK.
* **Негативний випадок**: коли для finding не повинно бути знайдено зовнішньої цитати (наприклад, суто структурне правило без нормативного аналога – тест очікує, що resolve\_citation поверне None).
* **Формат виходу**: тести мають перевірити, що повернутий об’єкт є саме Citation (заповнений згідно схем), містить Evidence з непорожнім текстом, а score – число в допустимому діапазоні (0.0–1.0).

**Очікуваний результат:** Новий файл тестів, наприклад tests/test\_citation\_resolver.py, з набором тестових функцій. Кожен тест створює мінімальний об’єкт Finding (з необхідними полями, як то rule або description) і викликає resolve\_citation. Потім за допомогою assert перевіряє результат. Будуть як мінімум такі тести:

* Тест на кейс з GDPR/ICO: наприклад, Finding про персональні дані -> очікувано Citation.source == "legislation" (або "ICO"), title містить "UK GDPR", evidence.text містить фрагмент про *“processor must obtain a commitment of confidentiality…”*.
* Тест на кейс OGUK: Finding з правилом, що містить "Oil&Gas" -> очікувано Citation.source == "OGUK", title містить "OGUK", evidence.text містить певний пояснюючий текст, score == 1.0.
* Тест на відсутність цитати: Finding типу "Definitions missing" -> очікувано функція поверне None (або Citation з мінімальними полями, якщо такий дизайн).

Всі тести повинні проходити (pytest), використовуючи реалізацію алгоритму. Очікуваний diff – доданий файл з тестами.

**Codex prompt:**

Create a new test module `tests/test\_citation\_resolver.py` for unit testing the citation\_resolver.

Include the following tests:

- \*\*Personal Data Clause Missing\*\*: Simulate a Finding indicating a missing personal data processing clause. For example, create a Finding with rule or description containing "personal data". Call `resolve\_citation(finding)` and assert that:

- The result is not None.

- `result.citation.title` (or equivalent field) contains "UK GDPR" (expecting a reference to UK GDPR Article 28(3)).

- `result.evidence.text` contains the snippet "the processor must obtain a commitment of confidentiality" (confirming the correct evidence was fetched).

- `result.score` is a float (e.g., 1.0).

- \*\*Oil & Gas Contract\*\*: Simulate a Finding for an Oil&Gas contract rule. E.g., `finding.rule = "Oil&Gas Master Agreement missing section"`. Expect that `resolve\_citation` returns a Citation with `source` indicating "OGUK", and `citation.title` contains "OGUK". Verify the evidence text is not empty and `result.score == 1.0`.

- \*\*No Citation Available\*\*: Simulate a Finding with no applicable external source (e.g., `finding.rule = "Definitions"` for a missing Definitions section). Assert that `resolve\_citation` returns `None` (or that the citation field is None/empty, depending on implementation).

Use Pydantic models from core/schemas to construct Finding objects (you can minimalistically fill required fields). Ensure all assertions pass.

Make sure tests are formatted with pytest style and cover the expected behavior thoroughly. Provide the test code as a diff.

**5. Інтеграція з pipeline**

**Опис задачі:** Після реалізації блоку citation\_resolver його потрібно інтегрувати у загальний конвеєр обробки документів. Pipeline наразі виглядає як: **DOCX → rules → findings → (тут вставити citation\_resolver) → Word (результат)**. Треба внести зміни у той компонент pipeline, який формує вихідні результати (ймовірно, функція аналізу або основний модуль на кшталт contract\_ai\_doctor.py чи аналог). На етапі після отримання списку Findings необхідно:

* Ітеруватися по кожному Finding і викликати resolve\_citation для нього.
* Отриманий Citation (якщо не None) прикріпити до відповідного Finding. Якщо в моделі Finding вже є поле citation, присвоїти туди об’єкт Citation. Якщо поле – список, додати до списку.
* Забезпечити, що модифікований Finding все ще відповідає схемі (наприклад, пройде валідацію Pydantic). Після додавання цитат вихідний формат результату / API не повинен порушитись.
* Прокинути ідентифікатор запиту (cid) в логування citation\_resolver (можливо, як параметр або через контекст), щоб у логах зберегти зв’язок між викликом та знайденими цитатами.

**Очікуваний результат:** Зміни в модулі pipeline (наприклад, в engine/pipeline.py або в місці, де збирається список Findings перед поверненням користувачу). У diff це може виглядати як доданий цикл над findings із викликом resolve\_citation і присвоєнням finding.citation = .... Також, можливо, імпорт нового модуля (from core import citation\_resolver). Інтеграцію слід розробити так, щоб при відсутності цитати поле залишалося порожнім/None, а при наявності – додавалося. Після цього етапу pipeline повинен віддавати повноцінні Findings зі вставленими Citation, які відобразяться у Word-надбудові (наприклад, як примітки чи частина рекомендацій). Усі існуючі тести pipeline/API повинні залишитися валідними.

**Codex prompt:**

Integrate the citation\_resolver into the analysis pipeline.

Instructions:

1. Open the module that orchestrates the pipeline (e.g., `engine/pipeline.py` or similar where findings are collected).

2. Import the resolve\_citation function (e.g., `from core.citation\_resolver import resolve\_citation`).

3. After the step where findings are generated by rules (a list of Finding objects), loop through each finding and call `citation = resolve\_citation(finding)`.

4. If a citation is returned (not None), attach it to the finding. For example: `finding.citation = citation` (assuming the Finding model has a field for citation as updated in schemas). If the schema uses a list, do `finding.citations.append(citation)`.

5. Handle the case of None (i.e., do nothing or ensure the field remains None).

6. Ensure that the output data structure still conforms to core/schemas.py (the Pydantic model validation should pass for the modified findings list).

7. Pass along or have access to the request’s `cid` (contract ID) if needed for logging inside resolve\_citation – for example, ensure that the logger or resolve\_citation can retrieve the current cid (this might already be available via a global context or the finding itself).

Make minimal changes required. All pipeline tests and integration tests must still pass after this update. Provide the diff of the pipeline integration changes.

**6. Безпека та логування**

**Опис задачі:** Необхідно впровадити заходи, що гарантують безпечну роботу блоку citation\_resolver у відповідності до вимог. Це включає:

* **Відсутність збереження даних (no-retention):** модуль не повинен зберігати ні в файл, ні в базу даних жодної інформації із вхідного документа або отриманих цитат. Всі обробки – лише в пам’яті, результат одразу повертається користувачу. Переконатися, що код не містить побічних ефектів типу кешування з конфіденційними даними.
* **Аудит-логування:** використати механізм логування для фіксації дій модуля, не порушуючи конфіденційності. Для кожного finding, що обробляється, створити запис у журналі (рівень INFO) з вказанням щонайменше CID запиту та стислого опису результату (наприклад: “CID123: Citation for rule ‘DataProtectionClauseMissing’ → UK GDPR Art28 found”). Це забезпечить аудитну трасу для відстеження роботи ІІ. Логи повинні бути інтегровані у загальний лог-файл системи.
* **Виключення PII з логів:** переконатися, що у повідомленнях логів **не міститься** жодних персональних чи чутливих даних із контракту. Логувати тільки службову інформацію: ідентифікатори, назви правил, посилання на нормативні акти. Якщо є ризик, що текст evidence може містити щось з оригіналу – не логувати текст цитати повністю.
* **Конфігурація Sentry/моніторингу:** якщо у проекті підключено Sentry або інший трасуючий сервіс для винятків, пересвідчитись, що новий код не передає туди PII. (Наприклад, не кидає винятків з частинами контракту; якщо виникає помилка при виклику зовнішнього API, логувати її без деталей запиту).
* **Перевірка безпеки:** провести самостійний аудит коду citation\_resolver після реалізації – переконатися, що він не додає вразливостей (наприклад, виконує запити до зовнішніх API по HTTPS, обробляє помилки мережі, не допускає SQL-ін’єкцій — хоча тут це малоймовірно).

**Очікуваний результат:** Відредагований код з доданими викликами logger (чи відповідного логера) там, де потрібно, та відсутністю зберігання даних. У diff це може бути помітно у функції resolve\_citation: додано кілька рядків logger.info(...) з потрібним повідомленням, а також, можливо, у конфігурації логування – вимкнено надмірний збір даних. Окремого файлу тут немає — зміни розподілені по існуючих (напр. в citation\_resolver та можливо в налаштуваннях). Усі зміни повинні пройти тести і лінтери.

**Codex prompt:**

Review and update the citation\_resolver implementation for security and logging best practices.

Tasks:

- Add logging: Use the project’s logging system (e.g., `loguru` or `logging`) to log an info message whenever a citation is resolved. Include the contract ID (cid) and the rule identifier from the Finding, and a brief note of the citation reference. For example: `logger.info(f"CID {cid}: resolved citation for rule {finding.rule} -> {citation.title} {citation.section}")`. Do NOT include any raw text from the contract or full evidence in the log to avoid PII.

- Ensure no data retention: The function should not write any files or persist data. Verify that all data (findings text, search results, etc.) are handled in-memory and discarded after use.

- Sanitize external calls: If the resolver makes any external API calls or uses external services, ensure it uses HTTPS and proper error handling. Do not log full request data if it contains parts of the document.

- Check that no exceptions leak sensitive info. If using Sentry (sentry-sdk) for error reporting, ensure that either it is configured to not capture local variables or no PII is in exceptions. (This might be configured globally, but double-check that our code doesn’t attach PII to exceptions or logs.)

- Document any important security consideration in comments, if helpful, e.g., “# NOTE: not logging the content to protect PII”.

Make minimal necessary changes to implement the above. Provide the diff for changes in `citation\_resolver.py` (and any other file, if logging config needs adjustment).

**Примітка:** Після виконання всіх задач новий блок citation\_resolver має безперебійно працювати в складі Hybrid RAG-системи. Він прийматиме Findings від правил, збагачуватиме їх точними юридичними посиланнями з потрібними метаданими та повертатиме у форматі, що відповідає схемам. Всі кроки повинні бути прозорими для аудиту, а обробка – безпечною (ніяких витоків даних). Ці інструкції призначені для команди розробників/DevOps, щоб реалізувати і деплоїти даний функціонал поетапно, гарантуючи якість (через тести і лінтери) та відповідність бізнес-вимогам.