прийнято. КРОК 7 — **Learning (локальна адаптація)**. Нижче — детальний, “промисловий” план, що ідеально лягає на нашу існуючу архітектуру (SSOT, rule-first, zero-copy spans, envelope {meta,data}), з акцентом на інноваційність, продуктивність, точність і приватність. Формат: **Architecture/Design → Plan (файли) → I/O → Алгоритм → Перф/Безпека → Tests → Validation → Next steps → STATE LOG.**

**Architecture / Design (єдина логіка)**

**Мета:** персоналізувати вибір шаблонів (draft/suggest) під користувача **без** зміни правових вердиктів (findings/risk/status).  
**Принципи (guardrails):**

* **Rule-first:** learning **не змінює** правила/ризики/статуси; лише **re-rank** шаблонів і стилістичні параметри в межах mode.
* **Локально/приватно:** події й ваги — лише на диску сервера; **жодних** зовнішніх відправок.
* **Детермінізм:** однакова історія ⇒ однаковий порядок шаблонів.
* **Безпечне зберігання:** ротація, atomic write, file-lock, HMAC підпис рядків JSONL.
* **Контрольованість впливу:** final\_score має **clamp** (обмеження зміни відносно base\_score).

**Plan (файл → що створюємо → входи/виходи → Done when)**

**1) contract\_review\_app/learning/replay\_buffer.jsonl (новий, append-only журнал)**

**Що:** події “прийняв/відхилив/accept all/reject all” по клауза/шаблону.  
**Рядок JSONL (ASCII-only):**

{

"schema\_ver": "1",

"event\_id": "uuid4",

"event\_key": "sha256(doc\_id|clause\_id|template\_id|action|ts\_minute|salt)",

"ts": "2025-08-15T14:12:03Z",

"user": "local",

"user\_role": "buyer|seller|neutral",

"doc\_id": "sha256(text|salt)",

"clause\_id": "Clause.id",

"clause\_type": "confidentiality|jurisdiction|...",

"contract\_type": "MSA|NDA|SaaS|other",

"mode": "friendly|standard|strict",

"action": "applied|rejected|accepted\_all|rejected\_all",

"template\_id": "CONF\_TPL\_03",

"suggestion\_id": "optional",

"span": {"start": 123, "length": 45},

"context": {

"jurisdiction": "UK",

"language": "en",

"policy\_pack": "default@1",

"risk\_ord": 2,

"severity\_max": "high"

},

"proposed\_text\_hash": "sha256(text|salt)",

"verdict\_snapshot": {

"status\_from": "WARN",

"status\_to": "OK",

"risk\_ord\_from": 2,

"risk\_ord\_to": 1,

"score\_delta": 8

},

"ui\_latency\_ms": 320,

"client": {

"cid": "x-cid",

"app\_build": "api-20250815",

"panel\_build": "build-20250815-1200"

},

"hmac": "hex(signature(line\_without\_hmac))"

}

**Вхід/вихід:** append рядків; ротація/архів.  
**Done when:** файл створюється, дублікати (за event\_key) не проходять, ротація працює.

**2) contract\_review\_app/learning/adaptor.py (новий, логіка)**

**Публічні функції (стабільні сигнатури):**

* get\_config() -> dict — {enabled, retention\_days, rotation\_mb, update\_min\_events, version}
* log\_event(event: dict) -> None — валідація, dedup по event\_key, file-lock, HMAC, append, ротація.
* update\_weights(min\_events: int | None = None) -> dict — стрім-агрегація нових подій (за watermark), Laplace+EWMA, clamp, atomic write weights.json; повертає {updated, events\_used, template\_count}.
* rank\_templates(clause\_type: str, context: dict) -> list[dict] — [{template\_id, score, reason}] за сегментом ключа.

**Вхід/вихід:** JSON події ↔ JSON ваги.  
**Done when:** при відсутності подій повертаються дефолт-ваги; при наявності — стабільний re-rank.

**3) contract\_review\_app/learning/weights/weights.json (новий, артефакт)**

**Структура:**

{

"version": "1",

"updated": "2025-08-15T14:15:09Z",

"watermark": {"last\_event\_ts": "2025-08-15T14:14:59Z", "last\_event\_id": "uuid4"},

"by\_segment": {

"(clause\_type=confidentiality|mode=strict|jurisdiction=UK|contract\_type=NDA|role=buyer)": {

"CONF\_TPL\_01": {"score": 0.62, "n": 31, "trend": "+", "last": "2025-08-15T14:13:01Z"},

"CONF\_TPL\_03": {"score": 0.41, "n": 18, "trend": "="}

}

}

}

**Done when:** atomic write (.tmp → rename), читається без помилок.

**4) contract\_review\_app/learning/README.md (новий, політика)**

**Що:** opt-in/opt-out, що логимо/не логимо, salt/HMAC ключ (локальний файл), як стерти історію, як вимкнути.

**5) contract\_review\_app/api/app.py (додати маршрути Learning)**

* POST /api/learning/log (batch) → body: {"events":[...≤100]} → {"status":"ok","accepted":N}
* POST /api/learning/update → {"min\_events":50} → {"updated": "...", "events\_used": 128, "template\_count": 23}
* Ті самі envelope/meta заголовки: x-cid, x-schema-version, Problem+JSON на помилках.  
  **Done when:** події пишуться; update повертає валідні метрики.

**6) Інтеграція в генерацію драфтів/пропозицій**

**engine/pipeline.py → synthesize\_draft(analysis\_or\_dict, mode)**  
Перед вибором шаблонів:

* побудувати segment\_key за контекстом (clause\_type, mode, jurisdiction, contract\_type, user\_role),
* adaptor.rank\_templates(...) → re-rank **в межах mode**,
* застосувати **clamp**: final\_score = 0.7\*base + 0.3\*learned, |final - base| ≤ 0.25.  
  Додати в метадату драфта: learning: {template\_id, score, events\_used}.

**gpt/gpt\_draft.py (коли LLM on):**  
Після rule-шаблонів, до LLM-стилізації — той самий re-rank; guardrails лишаються (не знижує rule-severity, не вигадує норми).

**Done when:** при однаковому вводі й наявних вагах — детермінований порядок варіантів; без learning — дефолт-порядок.

**7) Панель (taskpane.bundle.js) — мінімальна проводка**

* На **Apply (tracked)** / **Reject** → POST /api/learning/log (batch, по факту підтвердженої дії), **без** plain-text; тільки хеші/ID + метрики з /api/qa-recheck (якщо є).
* У Doctor показувати: learning: enabled, weights updated@, events\_used.  
  **Done when:** події доходять у бек; статус видно в Doctor.

**I/O контракти (строго)**

**POST /api/learning/log**

**Request:**

{

"events": [

{

"doc\_id": "sha256",

"clause\_id": "abc123",

"clause\_type": "confidentiality",

"contract\_type": "NDA",

"mode": "strict",

"user\_role": "buyer",

"action": "applied",

"template\_id": "CONF\_TPL\_03",

"span": {"start": 100, "length": 40},

"verdict\_snapshot": {"status\_from": "WARN", "status\_to": "OK", "risk\_ord\_from": 2, "risk\_ord\_to": 1, "score\_delta": 8}

}

]

}

**Response:** {"meta":{cid,...},"data":{"status":"ok","accepted":1}}

**POST /api/learning/update**

**Request:** {"min\_events":50} (optional)  
**Response:** {"meta":{...},"data":{"updated":"...","events\_used":128,"template\_count":23}}

**adaptor.rank\_templates(clause\_type, context) -> [{template\_id, score, reason}]**

context містить: mode, jurisdiction, contract\_type, user\_role, policy\_pack.

**Алгоритм ваг (v1, швидко/надійно/пояснювано)**

**Сегментація:**  
segment\_key = (clause\_type, mode, jurisdiction, contract\_type, user\_role)

**Лічильники з Laplace (α=1):**  
A = 1 + #applied, R = 1 + #rejected ⇒ p = A / (A + R)

**EWMA (λ=0.2) по часу подій:**  
learned\_score\_t = 0.8 \* learned\_score\_(t-1) + 0.2 \* p\_t

**Бонус за якість (малий, clamp ±0.02):**

* якщо risk\_ord\_to < risk\_ord\_from → learned\_score += 0.02
* якщо score\_delta > 0 → learned\_score += 0.01

**Фінал з clamp впливу:**  
final\_score = 0.7 \* base\_score + 0.3 \* learned\_score  
|final\_score - base\_score| ≤ 0.25

**Тай-брейк:** при рівності — за template\_id ↑.

**Watermark:** зберігати last\_event\_ts/id у weights.json щоб інкрементально рахувати.

**Performance / Security / Privacy**

**Перф:**

* update\_weights: 10k подій ≤ 300 ms (стрім-парсинг, без повного завантаження в пам’ять).
* rank\_templates: ≤ 2 ms/виклик (читання маленького weights.json у пам’яті, кеш на процес).

**Безпека/приватність:**

* **Немає** plain-text; усі хеші з **per-machine salt** (файл learning/.salt) ⇒ офлайн-зіставлення неможливе.
* **HMAC** підпис кожного рядка JSONL (ключ у learning/.key), пошкоджені рядки — ігнор.
* File-lock під час запису/оновлення; **atomic rename** для weights.json.
* Ротація: replay\_buffer.jsonl → .gz при rotation\_mb та retention\_days.

**Tests**

**Unit (adaptor):**

* log\_event відсікає дубль по event\_key; валідує обов’язкові поля; додає HMAC; ротує файл.
* update\_weights на штучних подіях: росте score для “applied”; trend відображає останні зміни.
* rank\_templates без ваг → дефолтний порядок; з вагами → очікуваний re-rank; clamp працює.

**Property:**

* При 50/50 applied/rejected → learned\_score ~ 0.5 ± ε.
* При суцільних applied → → 1.0 з обмеженням впливу на final\_score.
* Видалення подій поза retention\_days змінює ваги очікувано.

**E2E:**

* Панель: 3× Apply одного шаблону → наступний драфт під цю клаузу ставить його першим (видно у метаданих драфту learning.template\_id).
* /api/learning/update повертає events\_used > 0, template\_count ≥ 1; Doctor показує “learning: enabled, updated@time”.

**Fail-safety:**

* Пошкоджений рядок JSONL → пропускається, процес не падає.
* Конкурентні log\_event → жодних гонок, файл цілісний.

**Validation (приймання)**

* **Rule-first захист:** жодна навчальна зміна не впливає на findings/risk/status.
* **Детермінізм:** при однаковій історії — однаковий порядок шаблонів (в т.ч. тай-брейк).
* **Приватність:** ніде не зберігаємо повні тексти; лише хеші + salt.
* **Продуктивність:** рангування ≲ 2 ms, оновлення ваг ≲ 300 ms на 10k подій.
* **UX:** в панелі видно статус learning; після кількох дій користувача результат “наближається”.

**Next steps (виконуємо по одному файлу)**

1. Створити каркас: learning/replay\_buffer.jsonl (порожній), learning/weights/weights.json (порожня структура), learning/.salt, learning/.key, learning/README.md.
2. Реалізувати learning/adaptor.py (log\_event / update\_weights / rank\_templates) з lock/atomic/HMAC/rotation.
3. Додати маршрути в api/app.py: /api/learning/log (batch) і /api/learning/update.
4. Інтегрувати rank\_templates(...) у pipeline.synthesize\_draft і gpt/gpt\_draft.py (із clamp і метаданими у відповіді).
5. У taskpane.bundle.js викликати /api/learning/log після **успішного** Apply/Reject; в Doctor показувати статус learning.
6. Прогнати тести (unit/property/e2e), зафіксувати SLO та приймання.

**STATE LOG**

**Цілі:** локальна персоналізація без ризику для правової точності; приватність; детермінізм; реальна користь юристу.  
**Артефакти:** план файлів learning/\*, контракти API, алгоритм Laplace+EWMA+clamp, інтеграція у pipeline/gpt, події у панелі.  
**TODO (далі):** починаємо з **adaptor.py** (дам drop-in після твоєї “ОК”), потім API й проводка у pipeline/gpt та панелі.  
**Ризики:** дублікати/гонки/псування файлів — знято через event\_key, file-lock, HMAC і atomic write; bias від випадкових кліків — знято Laplace+EWMA+min\_events+clamp.