# Blackletter — BMAD V4 Implementation Guide (One-Stop)

**Project**: Blackletter — GDPR Processor-Obligations Checker
**Audience**: Vibe CEO, PM, Architect, UX, PO, SM, Dev, QA
**Scope**: A single, step-by-step guide that stitches together all artifacts produced so far (Analyst → UX → Architect → PO → SM) and shows exactly how to execute the IDE stage with agents until MVP ships.

---

## 0) TL;DR Quickstart (15 minutes)

1. **Clone & run** the scaffold (Windows):

   ```
   scripts\ps\dev.ps1
   ```

2. **Seed config** (already provided): `core-config.yaml` (LLM off, caps, OCR off).
3. **Shard docs** (creates per-section files for agents):

   ```
   *shard-doc docs/prd.md prd
   *shard-doc docs/architecture.md architecture
   ```

4. **Start new chat tabs** per agent in your IDE.
5. **Scrum Master** → open **SM Story Pack v1** and choose **Story 1.1**.
6. **Dev** builds 1.1, marks ready; **QA** verifies; loop stories **1.1 → 1.2 → 1.3 → 2.1 → 2.2 → 3.1 → 3.2 → 2.4 → 4.1 → 4.2 → 5.1 → 5.2**.
7. **Export** a report from Findings; review Metrics wall; done.

---

## 1) What is a Story, and How the IDE Stage Works

- A **story** is a **testable feature slice** with: ID, epic, title, acceptance criteria, tasks, and tests.
- **IDE stage** = planning docs become **code** through an agent loop: 1) **SM**: turns approved PRD items into **developer-ready packets** (already prepared in the Story Packs).
  2) **Dev**: implements a single story using those packets + dev-load-always files.
  3) **QA**: verifies acceptance criteria, runs tests, may refactor.
  4) Repeat until epics complete.
- Always run each agent in a **fresh chat/tab** to keep context clean.

---

## 2) Artifact Map (what you already have)

- **Analyst (Mary)**: *Hand-Off Pack v1* — acceptance wordings, weak-language lexicon seed, rulepack skeleton, KPIs, gold-set plan.
- **PM (John)**: *PRD v1* — vision, epics, draft stories, KPIs, scope.
- **UX (Sally)**: *UI/UX Spec v1* — IA, flows, tokens, components, wireframes, v0/Lovable prompts.

- **Architect (Winston)**: *Architecture v1* — stack pins, source tree, API, detection engine, Windows commands.
- **PO (Sarah)**: *PO Validation & Approvals v1* — checklist pass, API/schema freeze, story approvals, sharding plan.
- **SM (Sam)**: *SM Story Pack v1* — dev-ready packets for 1.1, 1.2, 2.1, 2.2, 3.1.
- **Gap Closure Pack (Agents Complete v1)**: remaining stories (1.3, 2.4, 3.2, 4.1, 4.2, 5.1, 5.2), full rulepack a–h, schema validation, gold set + scorer, core config, error codes, PII redaction, PS scripts, CI, QA checklists.

Keep these under `docs/` and referenced in IDE chats.

---

## 3) Repository Layout (from Architecture v1)

```
blackletter/
  apps/
    web/                      # Next.js 14 app
    api/                      # FastAPI service
  docs/
    prd.md
    architecture.md
    prd/                      # (after sharding)
    architecture/             # (after sharding)
    stories/
  core-config.yaml            # authoritative config (LLM off, caps, OCR
off)
  scripts/ps/                 # Windows run/test/lint scripts
  versions.lock.md            # version pin record
  .github/workflows/          # minimal CI
```

---

## 4) Configuration (authoritative)

`core-config.yaml` (already provided) controls:
- **LLM**: `provider: none`, `gate_policy: snippet_only`, `snippet_max_tokens: 220`
- **Budget**: `hard_cap_tokens_per_doc: 1500`, `on_exceed: needs_review`
- **Cache**: SQLite keyed by `(prompt_id, snippet_hash)`
- **OCR**: `enabled: false`
- **Security**: `redact_pii: true` (only applied if LLM enabled)

Dev/QA should never flip these mid-story unless the story explicitly says so.

---

## 5) Sharding (critical prep)

Why: agents load **only** what they need.
Commands (BMAD task or CLI):

```
*shard-doc docs/prd.md prd
*shard-doc docs/architecture.md architecture
```

Expected structure:

```
docs/prd/epic-1.md …  docs/architecture/tech_stack.md
                      docs/architecture/coding_standards.md
                      docs/architecture/source_tree.md
                      docs/architecture/api_contracts.md
```

Add the 3 architecture shards + `core-config.yaml` + rulepack files to **dev-load-always**.

---

## 6) Agents & Chats (how to drive them)

- **SM chat**: open *SM Story Pack v1* (and Gap Closure Pack) → select the next **Approved** story.
- **Dev chat**: load **that story only** + dev-load-always files → implement.
- **QA chat**: load the same story + repo → verify and mark **Done**.

**Do**: one story per Dev chat, keep logs, commit often.
**Don't**: lump multiple stories into one change.

---

## 7) Sprint 1 Plan (order & success)

**Order** 1. **1.1** Upload & Job Orchestration
2. **1.2** Text Extraction (PDF/DOCX)
3. **1.3** Evidence Window Builder
4. **2.1** Rulepack Loader (art28_v1)
5. **2.2** Detector Runner (verdict + evidence)
6. **3.1** Findings Table (UI)
7. **3.2** Report Export (PDF/HTML)
8. **2.4** Token Ledger & Caps
9. **4.1** Metrics Wall
10. **4.2** Coverage Meter
11. **5.1** Org Settings
12. **5.2** Minimal Auth & Roles

**Sprint success** = Upload → Findings → Export works on baseline docs; Metrics shows tiles; Coverage = 8/8.

---

## 8) Story Packets (how to use)

Each packet has **Acceptance Criteria**, **Interfaces**, **Tasks**, **Tests**, and **Artifacts**.
- Dev follows tasks exactly; if architecture requires a change, raise to **SM/PO** first.
- QA mirrors **Acceptance Criteria** into a checklist and tests.

The full packets for missing stories (1.3, 2.4, 3.2, 4.1, 4.2, 5.1, 5.2) are in *Gap Closure Pack*. The rest are in *SM Story Pack v1*.

---

## 9) Detection Engine & Rulepack

- **Rulepack**: `apps/api/blackletter_api/rules/art28_v1.yaml` — detectors **(a)–(h)** with anchors, weak cues, and red flags.
- **Lexicon**: `rules/lexicons/weak_language.yaml` — hedges & vagueness.
- **Verdict mapping**: Pass (anchor + no red-flag) / Weak (anchor + hedge) / Missing (no anchor or contradicted) / Needs_review (ambiguous/over budget).
- **Schema validation**: JSON Schema + Pydantic in *Gap Closure Pack*.

**Unit tests**: minimum 3 **positive** + 3 **hard negative** snippets per detector.

---

## 10) Gold Set & Scoring

- **Corpus**: 12–20 contracts (public/synthetic).
- **Labels**: JSONL per finding with verdict + offsets.
- **Scorer CLI**: `tools/score_goldset.py` calculates per-detector Precision/Recall + macro averages; surfaces p95 latency, tokens/doc, %LLM.

**Gates**: Precision $\geq$ 0.85; Recall $\geq$ 0.90; Explainability $\geq$ 95%; p95 $\leq$ 60s; $\leq$ £0.10/doc.

---

## 11) Frontend UX (what to build)

- **New Analysis**: drag-drop uploader; stepper (Queued→Extracting→Detecting→Reporting→Done); clear error banners.
- **Findings**: evidence-first table; Verdict chips; Evidence Drawer with snippet + rule id + offsets; filter by verdict; search snippets.
- **Export**: dialog → PDF/HTML; branding + disclaimers.
- **Metrics (Admin)**: tiles (p95, tokens/doc, %LLM, explainability) + sparklines.
- **Settings**: LLM provider toggle (default none), OCR, retention; audit changes.
- **Accessibility**: WCAG AA, keyboard-first, visible focus; color not sole signal.

---

## 12) Security & Privacy

- LLM **off by default**; snippet-only when enabled; **PII redaction** before any LLM call.
- Signed URLs, MIME/size checks; retention default **none**.
- Export includes checksum and timestamp; UI and export show **not legal advice** disclaimer.

---

## 13) Running Locally (Windows)

- Start both services: `scripts\ps\dev.ps1`
- Run tests: `scripts\ps\test.ps1`

- Lint/format: `scripts\ps\lint.ps1`
- Web: `http://localhost:3000`
- API: `http://localhost:8000`

---

## 14) CI & Version Pins

- **CI**: minimal GitHub Actions for backend tests and frontend build/tests (included).
- **Pin policy**: record exact Python and npm trees in `versions.lock.md` on first commit; keep pins stable per sprint.

---

## 15) Error Codes & Microcopy (UI/API)

| Code | Message | Hint |
|---|---|---|
| `file_too_large` | File too large (max 10MB). | Compress or split the PDF. |
| `mime_unsupported` | Unsupported file type. | Upload PDF or DOCX. |
| `extract_failed` | Couldn't extract text. | File may be image-only; enable OCR in Settings. |
| `token_cap` | Token budget exceeded. | Some checks marked **Needs review**; reduce snippet size or raise cap. |
| `report_failed` | Report generation failed. | Retry; or download HTML export. |

Copy appears in toasts/banners and export footers.

---

## 16) Troubleshooting (agentic IDE)

- **Agent stuck**: start a new chat with only the current story + dev-load-always files.
- **Off-spec code**: stop, quote the acceptance criteria back to Dev; if needed, raise to SM to update the packet.
- **Performance drift**: check p95 on Metrics; profile extraction; confirm OCR is off.
- **False greens**: verify rulepack anchors; add hard negatives; widen red-flags.
- **Token cap hits**: confirm provider is **none**; if on, reduce snippet_max or raise cap deliberately.

---

## 17) Definition of Ready / Done (DoR / DoD)

- **DoR**: acceptance criteria written; artifacts named; dependencies listed; test data available; UX implications noted.
- **DoD**: unit + integration tests pass; acceptance criteria demo; QA note added; metrics updated; minimal docs/changelog updated.

---

## 18) Appendices

### A) API Contracts (freeze)

- `POST /api/contracts` → `{ job_id, analysis_id, status }`
- `GET /api/jobs/{job_id}` → job status
- `GET /api/analyses/{id}` → summary + coverage
- `GET /api/analyses/{id}/findings` → `Finding[]`
- `POST /api/reports/{analysis_id}` → `{ url }`

**Finding**

```
{
  "detector_id": "A28_3_c_security",
  "rule_id": "art28_v1.A28_3_c_security",
  "verdict": "pass|weak|missing|needs_review",
  "snippet": "…technical and organisational measures…",
  "page": 7,
  "start": 1423,
  "end": 1562,
  "rationale": "anchor present; no red-flag",
  "reviewed": false
}
```

### B) Commands Cheat Sheet

```
*help                         # list BMAD commands
*kb-mode                      # load knowledge base
*shard-doc <src> <outdir>     # shard markdown by H2
```

### C) Story Template (for new items)

```
id: <epic.story>
epic: <n>
title: <short>
status: draft|approved|review|done
acceptance_criteria:
  - <testable condition 1>
  - <testable condition 2>
interfaces:
  - <API/DB/Service contracts>
tasks:
  - <dev tasks>
tests:
  - <unit/integration/UI>
artifacts:
  - <files to create/modify>
```

## Final Note

This guide consolidates everything needed to execute **BMAD V4** for Blackletter. Keep agent chats **focused** (one story per tab), and let the documents lead the code. When in doubt, the acceptance criteria are the ground truth.