Job Fit & Salary Estimator

Design explainer — every step, every calculation, every decision

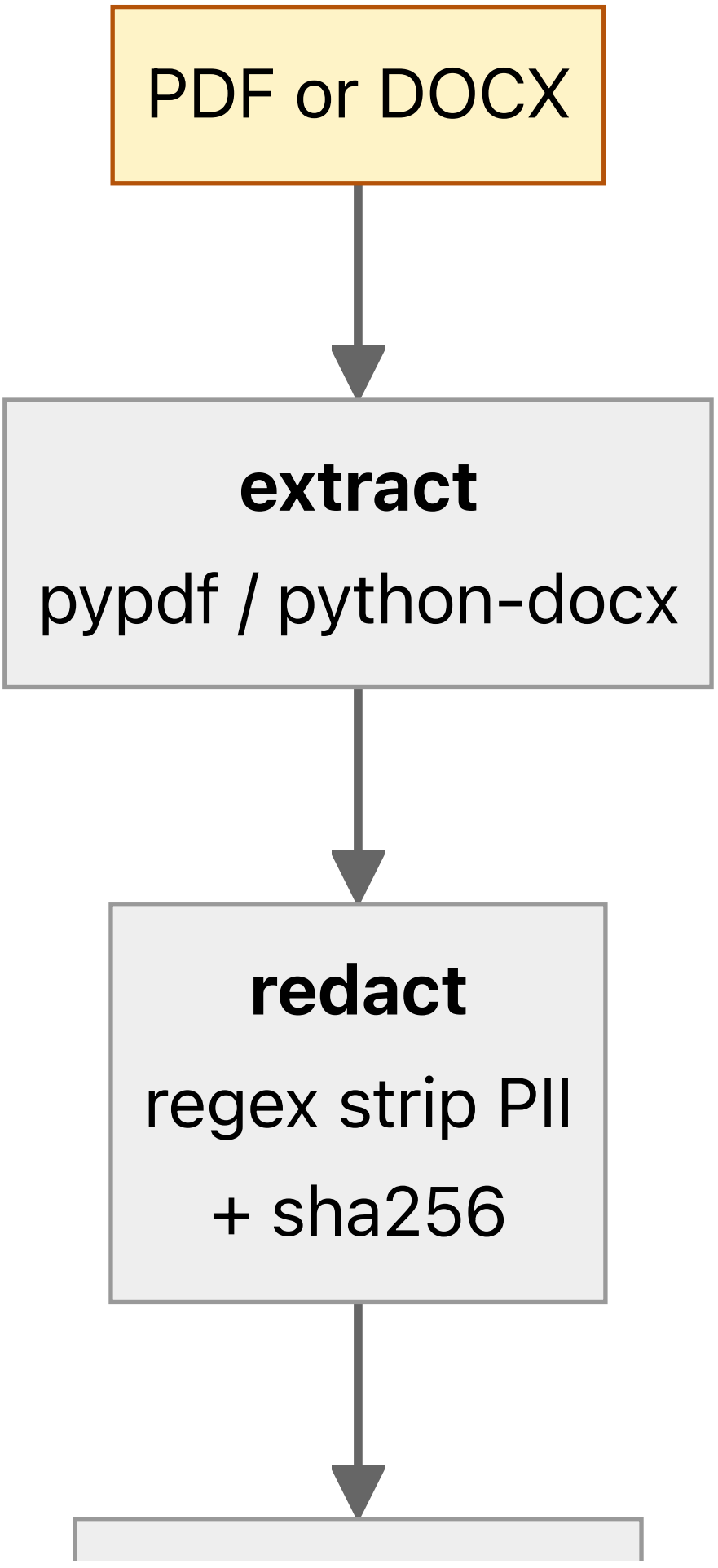
> ***Headline architecture:*** *Official wage anchor + transparent seniority model + bounded LLM explanation.*

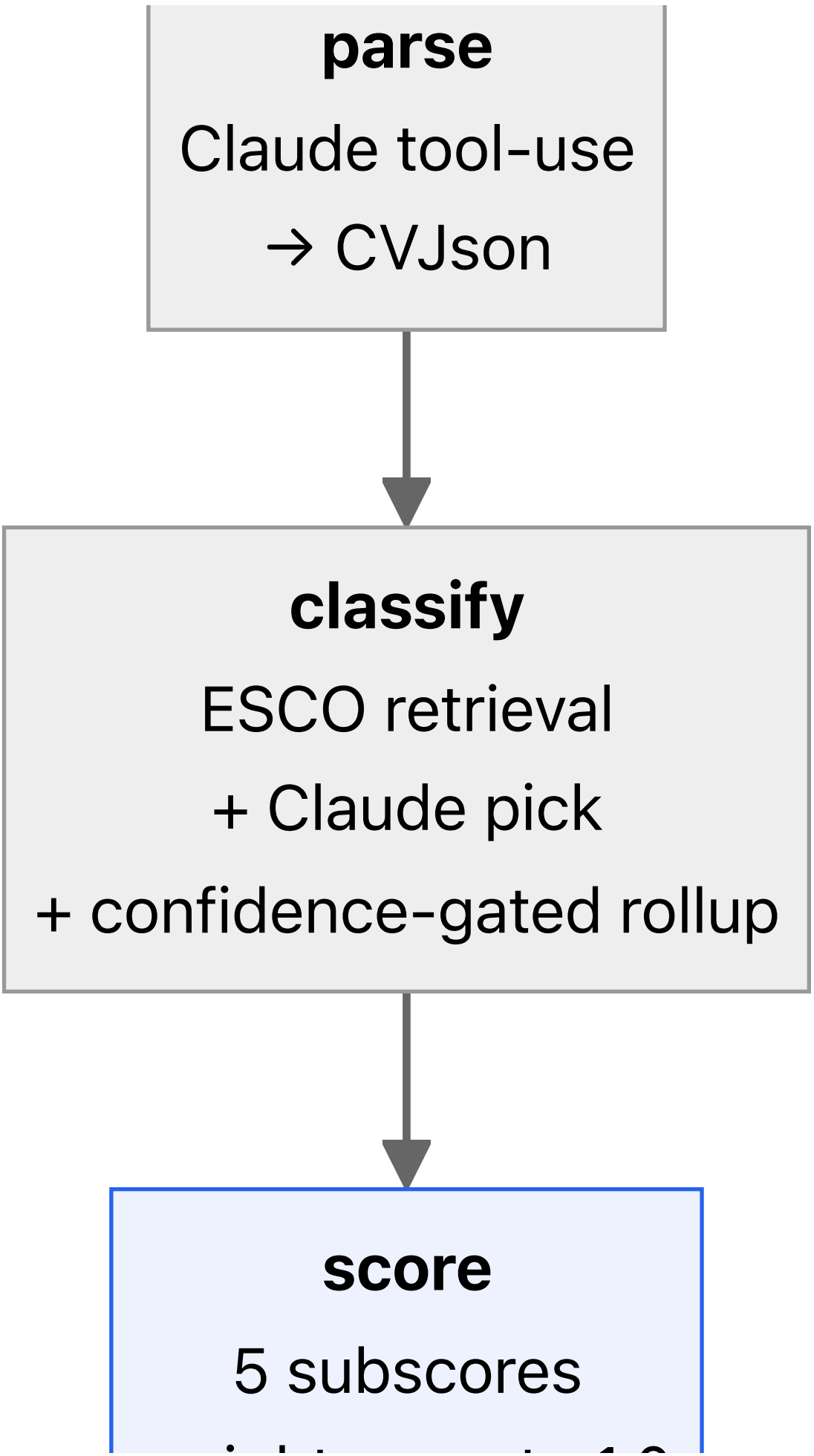
Version 0.1.0 · CV PDF/DOCX → seniority score 0–100 + salary range + +30% growth plan · Czech ISPV anchor · bilingual CZ/EN

CONTENTS

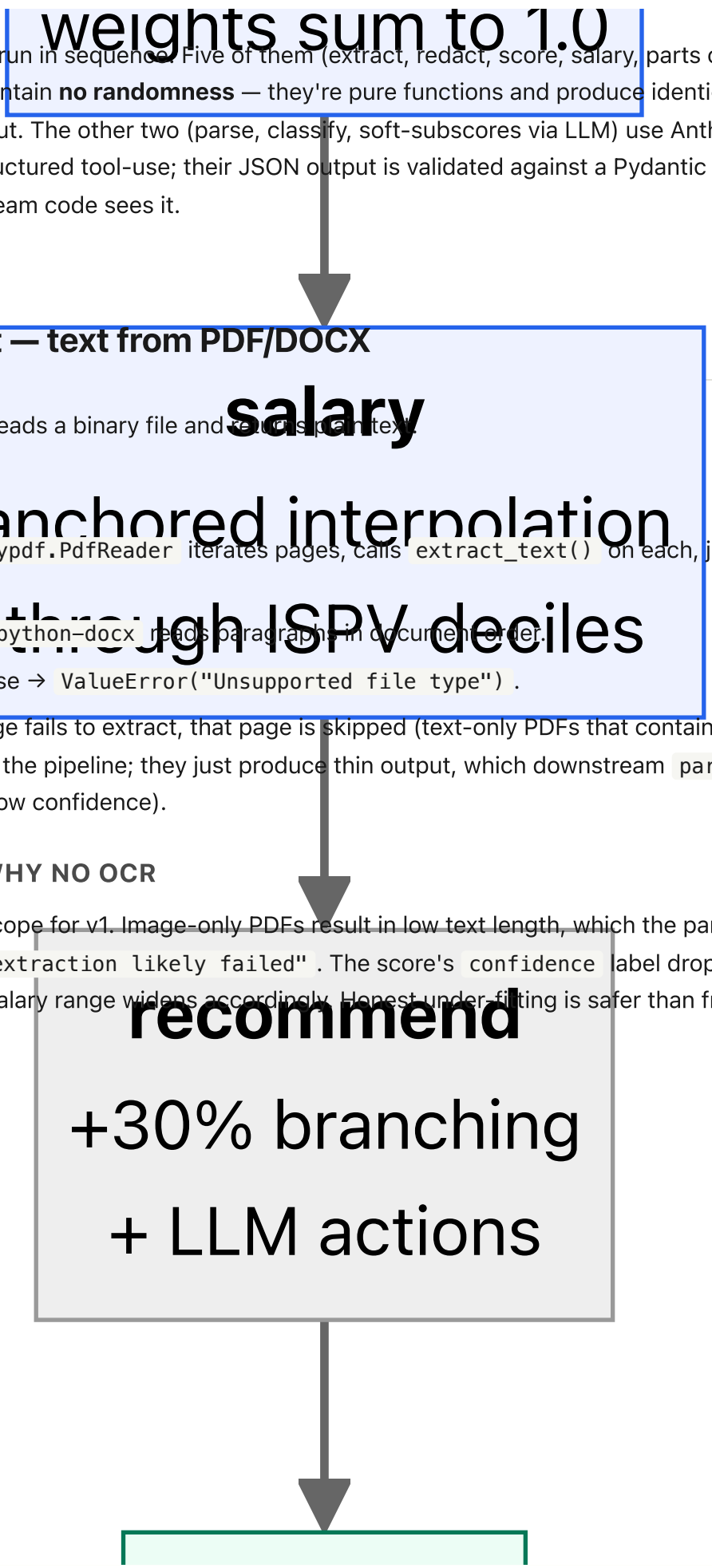
Pipeline overview

The system is a **pure-function pipeline with Pydantic-typed I/O at each step**. Every step is auditable in isolation: same input → same output. The LLM is bounded to evidence extraction and soft subscores within strict JSON schemas; final score and salary math are deterministic.


PDF or DOCX
extract
pypdf / python-docx
redact
regex strip PII
+ sha256


parse
Claude tool-use
→ CVJson
classify
ESCO retrieval
+ Claude pick
+ confidence-gated rollup
score
5 subscores

All seven steps run in sequence. Five of them (extract, redact, score, salary, parts of recommend) contain **no randomness** — they're pure functions and produce identical output for identical input. The other two (parse, classify, soft-subscores via LLM) use Anthropic Claude with structured tool-use; their JSON output is validated against a Pydantic schema before downstream code sees it.



1 Extract — text from PDF/DOCX

What it does: reads a binary file and returns plain text.

LOGIC

- `.pdf` → `pypdf.PdfReader` iterates pages, calls `extract_text()` on each, joins with newlines.
- `.docx` → `python-docx` reads paragraphs in document order.
- Anything else → `ValueError("Unsupported file type")`.
- If a PDF page fails to extract, that page is skipped (text-only PDFs that contain images don't crash the pipeline; they just produce thin output, which downstream `parse` reports as low confidence).

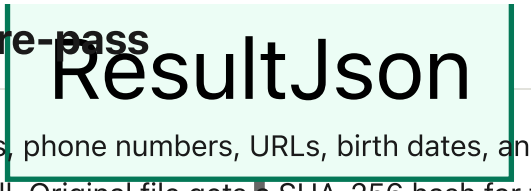
DECISION: WHY NO OCR

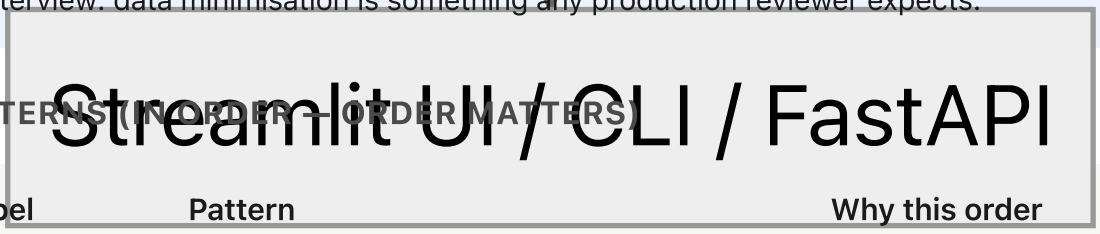
OCR is out of scope for v1. Image-only PDFs result in low text length, which the parse step flags as `"PDF extraction likely failed"`. The score's `confidence` label drops to `low`, and the salary range widens accordingly. Honest under-fitting is safer than fragile OCR.

2 Redact — PII pre-pass

What it does: strips emails, phone numbers, URLs, birth dates, and Czech ID numbers from the text *before* any LLM call. Original file gets a SHA-256 hash for traceability.

> **Why redact at all?** GDPR Art. 5 data minimisation. The system does not need names/contacts/addresses to estimate seniority and salary. Redacting before the LLM call means the model never sees them. This is also a defensibility win for the interview: data minimisation is something any production reviewer expects.





PATTERNS (ORDER = ORDER MATTERS)

Label	Pattern	Why this order
`BIRTH_DATE`	`(?:datum narození|date of birth|born|narozen[aá]?)\s*[:\-]?\s*\d{1,2}[./-]\d{1,2}[./-]\d{2,4}`	First — date numerals could otherwise be eaten by phone regex
`EMAIL`	`\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}\b`	Standard RFC-ish
`LINKEDIN`	`(?:https?://)?(?:www\.)?linkedin\.com/[^\s)]+`	Before generic URL — LinkedIn is more specific
`URL`	`https?://\S+|www\.\S+`	Generic fallback
`PHONE`	`(?:(?:\+|00)\d{1,3}[\s.-]?)?\d{3}[\s.-]\d{3}[\s.-]\d{3}(?!\d)`	Tightened to require explicit separators between three groups of 3 digits, so `03/2018` employment dates aren't captured
`CZ_ID`	`\b\d{6}\s?/\s?\d{3,4}\b`	Rodné číslo (Czech birth ID)

> **Known limitation:** regex doesn't catch names or addresses. Production would use NER (Named Entity Recognition). For an interview prototype, the regex pass + an

> explicit *"production would use NER"* note in the README is acceptable — the methodology is honest.

3 Parse — text → CVJson

What it does: Claude reads the redacted text and emits a structured `CVJson` via tool-use. The schema enforces the shape; Pydantic validates the result.

TOOL SCHEMA (ABBREVIATED)

```
{
  "name": "parse_cv",
  "input_schema": {
    "type": "object",
    "required": ["roles", "skills", "education", "languages", "certifications",
                 "detected_language", "parse_confidence", "extraction_warnings"],
    "properties": {
      "roles": {"type": "array", "items": {Role schema with title, dates, descrip
      "skills": {"type": "array", "items": {"type": "string"}},
      "education": {"type": "array", "items": {Education schema}},
      "languages": [...],
      "certifications": [...],
      "detected_language": {"enum": ["cs", "en", "other"]},
      "parse_confidence": {"type": "number", "minimum": 0, "maximum": 1},
      "extraction_warnings": {"type": "array", "items": {"type": "string"}}
    }
  }
}
```

DECISIONS

- **Self-reported confidence**: the LLM reports `parse_confidence ∈ [0,1]`. Below 0.6 it contributes a reason to the final score's confidence label.
- **Extraction warnings as tags**: short strings like `"no dates found"`, `"low text length"`, `"PDF extraction likely failed"`. These bubble up to the final ScoreCard's `confidence_reasons`.
- **Prompt caching**: the system block (instructions + extraction rules) is identical across CVs. Anthropic's prompt caching reads the cached block on subsequent calls, reducing input tokens by ~80% per CV after the first.
- **No invented data**: the system prompt explicitly forbids inventing roles, dates, or skills. Missing data → null + warning.

4 Classify — CV → ISCO with confidence-gated rollup

What it does: picks the best ISCO-08 occupation code for the candidate's anchor role. The output drives salary lookup downstream.

PROCESS

1. Pick anchor role (current role with longest duration; fallback to most recent role ≥ 12 months; fallback to most recent role).
2. Build a query string from anchor's title + description.
3. Query the local ESCO occupation index (28 hand-curated ISCO codes covering common occupations across all major groups) → top 8 candidates by Jaccard token overlap + substring bonus.
4. Pass the shortlist + CV skills + detected language to Claude. Claude picks ONE code, returns `{isco_code, role_label, confidence, top1_margin, alternatives}`.
5. Apply the **confidence-gated rollup**:

Confidence	top1_margin	ISCO level returned
≥ 0.80	≥ 0.15	`4` (4-digit, e.g. `2512`)
≥ 0.60	any	`3` (3-digit, e.g. `251`)
any	—	`2` (2-digit, e.g. `25`)

> **Why rollup matters:** a 4-digit ISCO has narrow salary distribution. If the LLM is unsure, returning the more aggregate 2-digit median is more honest than committing to a possibly-wrong 4-digit. Salary range automatically widens via the confidence label.

EDGE CASE: HIGH CONFIDENCE + TINY MARGIN

Confidence 0.85 with margin 0.05 means "I'm sure it's one of these two, I just don't know which." The gate returns level 3, not 4 — both candidates likely share a 3-digit prefix (e.g. `2511` Systems analyst and `2512` Software developer both roll up to `251`).

5 Score — five subscores blended

Five subscores, each 0–100, blended by fixed weights summing to 1.0:

Subscore	Weight	Computed by	Source
`relevant_experience`	0.25	Interval-union YoE × ISCO similarity	deterministic
`skills_match`	0.25	Keyword-group overlap with ISCO expected skills	deterministic
`impact_scope`	0.20	Numeric outcomes, scope of work — LLM-bounded with rubric	LLM-bounded
`leadership_ownership_growth`	0.20	5 sub-dims × 0–20, summed and capped — LLM-bounded	LLM-bounded
`education`	0.10	Role-sensitive ordinal mapping	deterministic

```
total = 0.25·rel_exp + 0.25·skills + 0.20·impact +
        0.20·leadership + 0.10·education
```

The total is mapped to a band:

Band	Total range
Junior	< 40
Mid	40 ≤ t < 60
Senior	60 ≤ t < 80
Lead/Principal	80 ≤ t < 95
Exec	95+

5a. `relevant_experience` (deterministic)

Two YoE measures, both with explicit handling of overlapping/parallel roles:

TOTAL YOE — INTERVAL UNION

Sum the union of all dated role intervals. Parallel jobs don't double-count.

```
def total_yoe(roles):
    intervals = [(r.start_date, r.end_date or today) for r in roles if r.start_da
    merged = merge_intervals(sorted(intervals))    # union: [(a, max(b, c)), ...]
    return sum(months_between(s, e) for s, e in merged) / 12.0
```

RELEVANT YOE — FTE-EQUIVALENT CAPPED

For each calendar month a role was active, contribute `isco_similarity × max(confidence, 0.5)`, capped at 1.0 per month so overlapping relevant roles can't sum to more than full-time. This is the correct way to handle freelance overlap.

```
def relevant_yoe(roles, anchor_isco):
    month_weights = {}
    for r in roles:
        weight = isco_similarity(r.isco_code, anchor_isco) * max(r.isco_confidenc
        for ym in iter_months(r.start_date, r.end_date or today):
            month_weights[ym] = min(1.0, month_weights.get(ym, 0) + weight)
    return sum(month_weights.values()) / 12.0
```

ISCO SIMILARITY

Match	Similarity	Example
identical 4-digit	1.00	2512 ↔ 2512
same 3-digit prefix (minor group)	0.75	2511 ↔ 2512 (both Systems analysts/Software devs)
same 2-digit prefix (sub-major)	0.50	2511 ↔ 2521 (both ICT professionals)
same occupation code at different level *(suffix match)*	0.25	2511 ↔ 3511 (Systems analyst Pro ↔ ICT technician)
unknown ISCO on either side	0.25	None ↔ 2512
different occupation entirely	0.10	2221 (Nurse) ↔ 2512 (Software dev)

> **Design note:** the suffix-match tier (0.25) was a deliberate fix. ISCO-08 is hierarchical by major-group (first digit), but the same occupation type at different seniority levels (Professional vs Technician) lives in *different* major groups (e.g. 2511 vs 3511). Strict prefix matching would say these are unrelated; suffix matching correctly says they're related. Conversely, two different occupations within the same major group (Nurse 2221 vs Software dev 2512, both "Professionals") are correctly scored as unrelated (0.10).

YOE → 0–100 SCORE (PIECEWISE INTERPOLATION)

Five anchor points calibrate years of relevant experience to a subscore:

Years	Score
0	0
2	30
5	55
10	80
15	95
20+	100

Linear interpolation between anchors. Concave (faster early gains, diminishing returns past 15y) — this matches market reality for IC roles.

PIPELINE GLUE: ISCO PROPAGATION TO ROLES

> **Subtle integration bug + fix.** The parse step doesn't classify each role individually — only the classify step picks ONE ISCO for the candidate. Without correction, every role has `isco_code=None`, so `isco_similarity` returns the 0.25 unknown-default for all of them, deflating relevant experience to roughly a quarter of its real value.
>
> The pipeline now propagates the classified ISCO to the anchor role at full confidence, and to any other role whose title shares a meaningful word with the classified `role_label` at 70% confidence. Career-changer roles (e.g. "Registered Nurse" vs "Software developer") have no shared title keywords, so they correctly stay None and contribute at the 0.25 baseline — preserving the property that career-changer's relevant YoE < total YoE.

5b. `skills_match` (deterministic, two-tier)

Three signals combined to 0–100:

- **Breadth** (0–40): `min(40, n × 6)` where `n` is the count of distinct skills extracted.
- **Depth** (0–30): substring hits of senior markers (`"architecture"`, `"system design"`, `"mentor"`, `"distributed systems"`, …) × 6, capped at 30.
- **Match** (0–30): overlap between extracted skills and the ISCO's expected keyword set, scaled.

TWO-TIER CONFIDENCE

- **Medium** (default for v1 core): keyword-group fallback, ISCO-2-digit-keyed dictionary.
- **High** (stretch, not in v1): ESCO occupation→skill mapping (Jaccard against expected skill set).
- **Low**: when the ISCO has no keyword set in the dictionary (e.g. truck drivers — fallback to neutral score).

5c. `impact_scope` (LLM-bounded)

The LLM scores 0–100 against this rubric, returns 1–4 evidence quotes from the CV:

Range	Description
0–20	Vague responsibilities, no measurable outcomes
21–40	Concrete deliverables, no numbers
41–60	Some measurable results (counts, percentages)
61–80	Business-level impact (revenue, cost, reliability, scale)
81–100	Cross-org / multi-million-scale impact

The schema enforces `integer 0–100`; Python clamps any out-of-range output as a safety net.

5d. `leadership_ownership_growth` (5 sub-dimensions × 0–20)

Originally called "personality" in the brief — renamed because CV text cannot defensibly infer personality. CV-observable signals are the honest proxy. Five dimensions, each scored 0/10/20 with evidence:

Dimension	0	10	20
Ownership	Task executor	Owns features	Owns outcomes / budgets
Leadership	No signal	Mentored / coordinated	Led people / strategy / hiring
Learning trajectory	Stagnant	Visible progression	Repeated upskilling / domain shifts
Impact clarity	Vague	Concrete deliverables	Measurable business results
Stability / execution	Unexplained hops	Normal transitions	Sustained delivery + promotions

Sum of the five = 0–100. Each clamped before summing as a safety net.

5e. `education` (deterministic, role-sensitive)

Highest formal degree → base score, then adjustment for relevance to the anchor ISCO:

BASE SCORE

None / unknown	30
High school / vocational	45
Bachelor's	65
Master's	80
PhD / Doctorate	90

RELEVANCE ADJUSTMENT

- +10 if degree field matches anchor ISCO 2-digit keyword set
- −20 if unrelated to anchor occupation (capped at floor 0)
- 0 otherwise

No institution-prestige scoring — deliberately excluded to avoid bias and the maintenance overhead of an "elite institution" allowlist. Reviewers see this in the README's "Limitations" section as a deliberate choice.

6 Salary math — the load-bearing logic

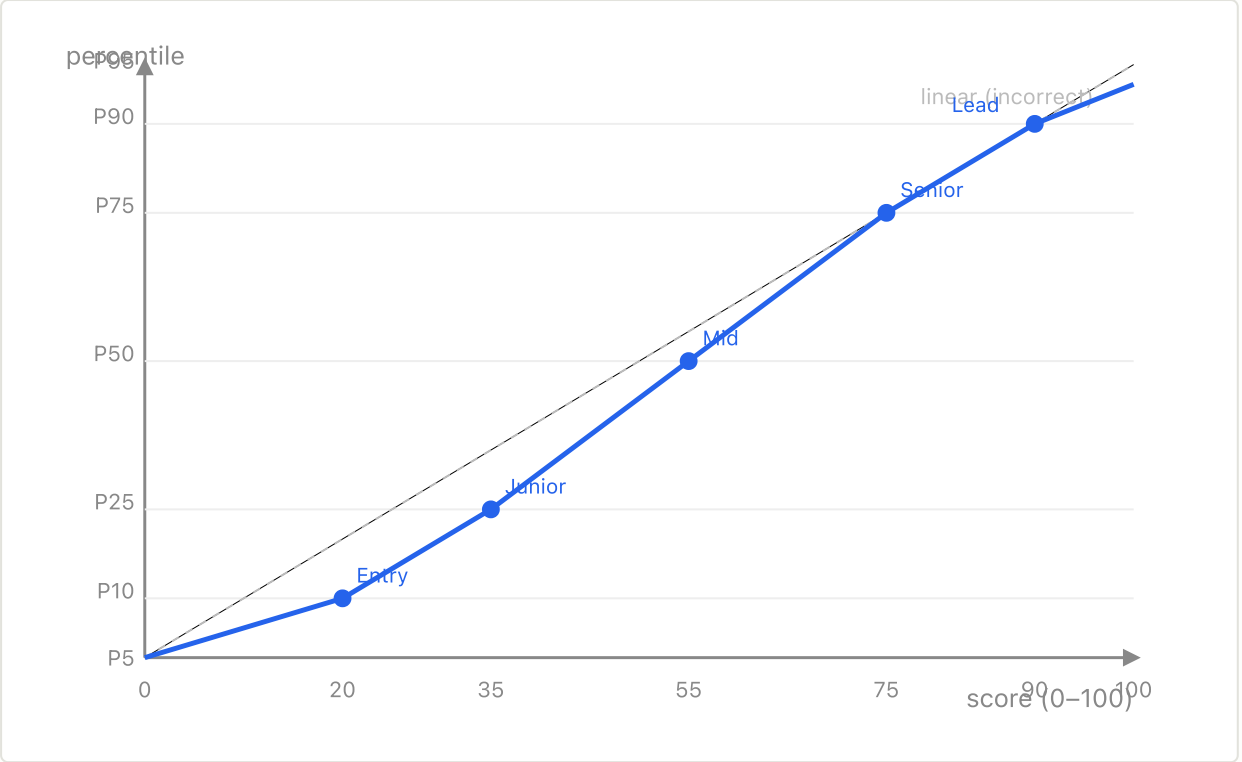
Two-stage interpolation: **score → percentile → salary**. The percentile axis is "where in your ISCO group's salary distribution does this candidate fall." The salary axis is the actual CZK figure read from MPSV ISPV for that ISCO.

6a. Score → percentile (anchored interpolation)

> **The most important design decision in the whole system.** A 75/100 candidate is *not* P75 of their ISCO cohort. The cohort already includes candidates from every score band; if 75/100 mapped to P75 linearly, the math would be silently wrong. We map *named seniority bands* to *realistic salary percentiles*:

Seniority band	Score anchor	Salary percentile
Entry / weak match	20	P10
Junior	35	P25
Solid mid	55	P50
Senior	75	P75
Lead / Principal	90	P90

Linear interpolation between anchors; clamped to P5–P95 at the edges (no extrapolation past observed deciles).



6b. Percentile → salary (linear through ISPV deciles)

MPSV ISPV publishes **D1, Q1, median, Q3, D9** per CZ-ISCO occupation. Linear interpolation between observed deciles, clamped to [P10, P90] — never extrapolates past observed data:

```
def percentile_to_salary(p, *, d1, q1, median, q3, d9):
    p = max(10, min(90, p))                    # clamp to observed range
    points = [(10, d1), (25, q1), (50, median), (75, q3), (90, d9)]
    for (p0, v0), (p1, v1) in zip(points, points[1:]):
        if p0 <= p <= p1:
            t = (p - p0) / (p1 - p0)
            return v0 + t * (v1 - v0)
```

Why no extrapolation past D1/D9: ISPV doesn't publish percentiles below D1 or above D9. Any "P95 of nurses" or "P3 of CEOs" would be invented, which is exactly the kind of fake precision the methodology is meant to avoid.

6c. Confidence-driven salary range width

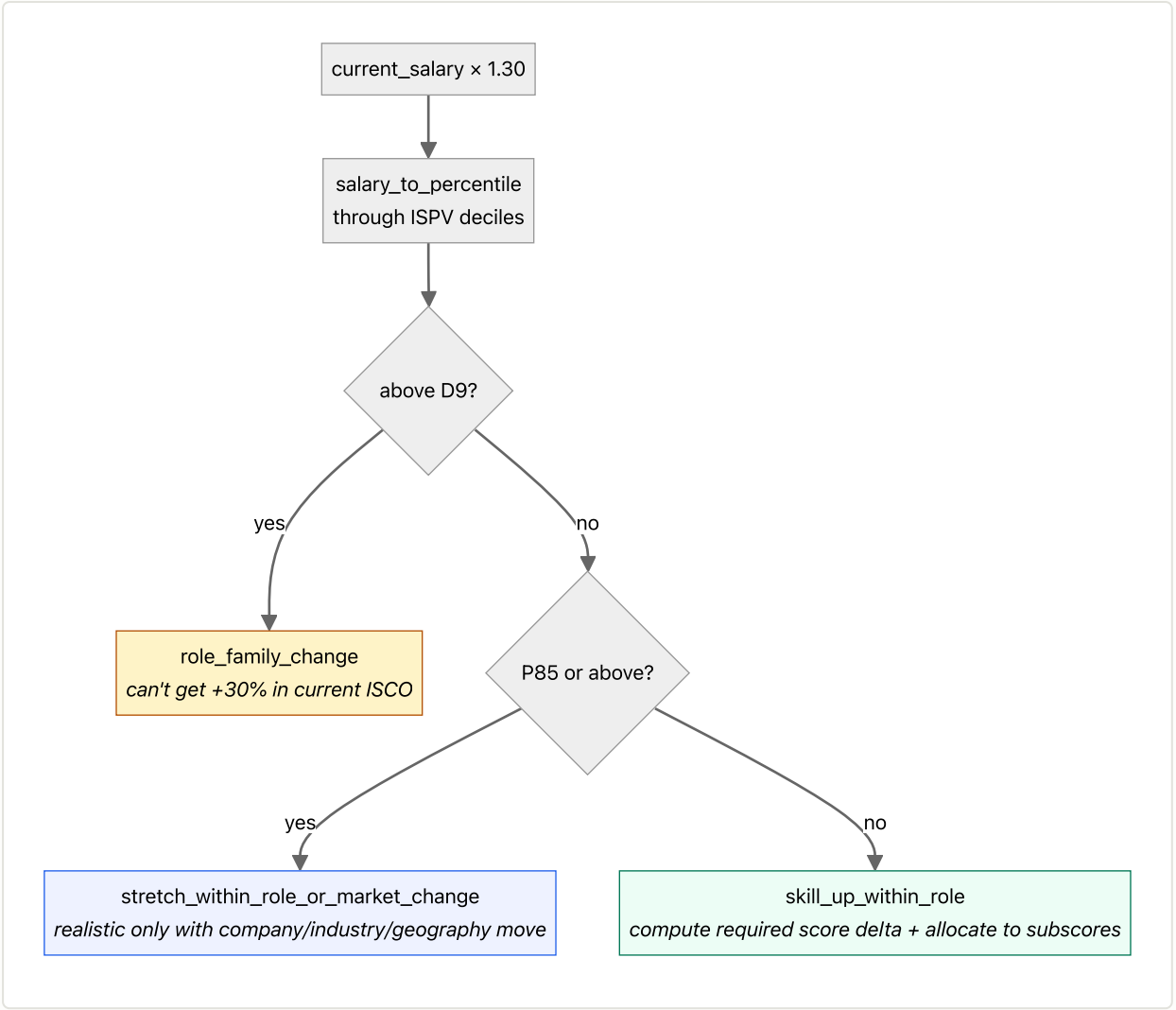
Confidence labels have a *mathematical* effect, not just a UI chip. Lower confidence widens the percentile band on each side of the point estimate before re-mapping to salary:

Confidence	± width (percentile points)	Effect on a P50 estimate
high	10	P40–P60
medium	15	P35–P65
low	25	P25–P75

Confidence label is `high` only with zero confidence reasons; it drops to `medium` on one reason, `low` on two or more.

7 +30% growth plan — three branches

Compute `target_salary = current_salary × 1.30`, find where it lands in the ISCO's salary distribution, and branch on that:



7a. `role_family_change`

Target salary exceeds the top decile of the candidate's current ISCO group. Even being P99 in this role won't get there. Recommendation is to **change occupation** (different ISCO, higher-paying industry) or **change geography/comp model** (foreign client, equity). Skill-up alone is mathematically impossible.

7b. `stretch_within_role_or_market_change`

Target sits in P85–P90. Mathematically reachable inside the current ISCO, but realistically requires a *combination* of demonstrated impact + a market move (changing companies, industries, or geographies). Skill-up alone usually isn't enough at this band.

7c. `skill_up_within_role`

Target lands inside P5–P85 of the current ISCO. Compute:

1. `target_percentile` from `salary_to_percentile(target_salary, ...)`
2. `required_score = percentile_to_required_score(target_p)` (inverse of the anchored interpolation)
3. `score_delta = required_score − current_score`
4. Allocate `score_delta` across subscores by **weighted capacity** (gap × weight), *skipping* `relevant_experience` — you can't fast-track years of experience.
5. Pass the deltas + redacted CV text to Claude → 3–5 concrete actions referencing CV evidence.

SUBSCORE-DELTA ALLOCATION

```
def allocate_subscore_deltas(subscores, required_total_delta, weights, skip):
    gaps = {k: 100 - subscores[k] for k in subscores if k not in skip}
    weighted_capacity = {k: gaps[k] * weights[k] for k in gaps}
    total_capacity = sum(weighted_capacity.values()) or 1.0
    plan = {k: 0.0 for k in subscores}
    for k in gaps:
        share = required_total_delta * (weighted_capacity[k] / total_capacity)
        plan[k] = min(gaps[k], share / weights[k])      # convert total-points ba
    return plan
```

> **Why this allocation rule:** the candidate gets the most "score per unit of effort" by investing in subscores that are both (1) low (large gap) and (2) heavily weighted in the total. Skills and impact dominate. Education's tiny 0.10 weight means a degree change rarely shows up as a recommendation.

Worked example: `mid_dev_4y.docx`

A 4-year-experienced software engineer in Czechia. The CV (paragraph form):

```
Jane Smith
Software Engineer

EXPERIENCE
Software Engineer — FinTech Plus, 05/2022 — present
- Owned the payment ingestion service handling 50k tx/day.
- Reduced p95 latency by 35% by introducing async Postgres pool.
- Mentored 1 intern.
Junior Developer — StartupX, 06/2020 — 04/2022
- Built REST APIs in Flask; helped migrate to FastAPI.
- Wrote integration tests; maintained CI pipeline.

SKILLS
Python, FastAPI, Flask, PostgreSQL, Redis, Docker, AWS, Kubernetes, Git, pytest, C

EDUCATION
Master's — Software Engineering, CTU Prague (2018–2020)
```

STEP-BY-STEP

Step	Output
1. extract	Plain text, ~600 chars (DOCX → string).
2. redact	No emails/phones/URLs in this CV → unchanged. SHA-256 of the original DOCX recorded.
3. parse	2 roles, 11 skills, 1 education entry. `parse_confidence = 0.95`. 3 minor warnings (no languages, no certs, education dates inferred).
4. classify	ISCO `2512` "Software developer". Confidence 0.95, top1_margin 0.75 → kept at 4-digit (no rollup).
5a. relevant_experience	Anchor role "Software Engineer" gets ISCO `2512`; non-anchor "Junior Developer" shares the word "developer" with role_label "Software developer" → gets `2512` at 0.7× confidence. relevant_yoe ≈ 4.0y · 0.95 + 1.83y · 0.665 ≈ **5.0y** → score ≈ **55**.
5b. skills_match	11 skills → breadth 40, no senior depth markers → 0, ISCO-25 keyword overlap (python, fastapi, postgresql, redis, docker, aws, kubernetes, git, pytest) → 30. Total **70**, confidence medium.
5c. impact_scope	"Reduced p95 latency by 35%", "50k tx/day" → measurable but not multi-million-scale → LLM scored **55**.
5d. leadership_ownership_growth	ownership 14 (owned ingestion service), leadership 10 (mentored 1 intern), learning trajectory 12 (Flask → FastAPI), impact_clarity 12 (concrete numbers), stability 12 (clean progression) → total **60**.
5e. education	Master's (80) + Software Engineering field matches ISCO 25 keywords (+10) → **90**.

Step	Output
total	0.25·55 + 0.25·70 + 0.20·55 + 0.20·60 + 0.10·90 = 13.75 + 17.5 + 11 + 12 + 9 = **63 → Senior**.
6. salary	Score 63 → percentile P60 (anchored interpolation). ISCO 2512 ISPV deciles: D1 ~50k, Q1 ~70k, median ~95k, Q3 ~130k, D9 ~180k. P60 ≈ **109,000 CZK/month**. Medium confidence (extraction warnings) → ±15 percentile points → range ~90,000–130,000 CZK.
7. growth plan	target = 109,000 × 1.30 = ~141,700 CZK. Salary→percentile ≈ P78 (within distribution but above P85 threshold? close). Branch: likely `stretch_within_role_or_market_change`. LLM generates 3–5 actions referencing CV evidence (e.g. "Lead a cross-functional initiative for 6 months and document business impact in numbers").

> **Note on numbers.** The exact figures above are illustrative — they reflect the methodology after the ISCO-propagation fix. The actual numbers from a live run will vary slightly with the LLM's exact subscore values for impact and leadership. The deterministic parts (relevant_experience, skills, education, salary math) are reproducible.

Validation

No labelled ground-truth dataset exists for "correct" CV scores. Instead, the system is validated by **rank-order assertions on a synthetic CV pack**: hand-crafted CVs at known seniority bands run through the full pipeline (live LLM calls), and the relative ordering must be sensible.

SYNTHETIC PACK (5 CVS IN V1 CORE)

CV	Profile	Expected band
`junior_dev_1y.docx`	1y dev, strong skills, low impact	Junior (25–40)
`mid_dev_4y.docx`	4y dev, normal progression	Mid (45–60) or Senior
`senior_dev_8y.docx`	8y dev, ownership, architecture	Senior (65–80)
`nurse_to_dev_5y_2y.docx`	Career changer (5y nurse → 2y dev)	Mid; salary anchors to dev ISCO not nurse
`buzzword_no_evidence.docx`	Verbose, no concrete results	Junior or low Mid

RANK-ORDER ASSERTIONS (LIVE E2E TESTS)

```
# tests/test_e2e_synthetic.py — gated on ANTHROPIC_API_KEY

assert score("junior_dev_1y") < score("mid_dev_4y") < score("senior_dev_8y")
assert score("buzzword_no_evidence") < score("mid_dev_4y")
assert relevant_yoe("nurse_to_dev_5y_2y") < total_yoe("nurse_to_dev_5y_2y")
assert classification("nurse_to_dev_5y_2y").isco_code.startswith("25")  # dev, no
assert score("senior_dev_8y").band in ("Senior", "Lead/Principal")
for cv in all_cvs: assert len(growth_plan(cv).actions) >= 3
```

Assertions are intentionally *loose* (band ranges, not exact scores) — the goal is verifying the methodology rank-orders correctly, not that the LLM produces specific numbers.

> **Latest run:** all 6 E2E rank assertions passed in 2:11 across 5 synthetic CVs (~20 LLM calls total). Plus 71 unit tests for the deterministic math (interval union, anchored interpolation, percentile↔salary inverse, +30% branching, etc.) running in < 1 second without an API key.

PURE-FUNCTION UNIT TESTS RUN WITHOUT AN API KEY

```
$ uv run pytest tests/ --ignore=tests/test_e2e_synthetic.py -q
71 passed, 1 skipped in 0.75s

$ uv run pytest tests/ --html=docs/reports/test-report.html \
    --cov=src/job_fit --cov-report=html:docs/reports/coverage --cov-report=term
TOTAL coverage: 86%
```

Data sources

Country	Source	Format	Refresh	Status
CZ	MPSV ISPV open data (`ispv-zamestnani.json`)	JSON	semi-annual	implemented
EU	Eurostat SES 2022 (`earn_ses_main`) via `eurostat` PyPI package	API	quadrennial	designed (stretch)
US	BLS OEWS + official SOC↔ISCO crosswalk	XLS/CSV	annual	designed (stretch)
UK	ONS SOC 2020 earnings + CASCOT mapping	CSV	annual	designed (stretch)
else	Eurostat fallback + explicit "low confidence" label	—	—	designed (stretch)

WHAT ISPV GIVES US

440 rows after fetch, period "rok 2025". Per CZ-ISCO occupation × wage/pay sphere (`MZDOVA` = private sector, `PLATOVA` = public sector), with fields: `medianMzda`, `diferenciaceD1M`, `diferenciaceQ1M`, `diferenciaceQ3M`, `diferenciaceD9M`, `mzdaPrumer`, `pocetZamestnancuMzda`, `obdobi`, `czIsco`.

Caveat: the public dataset is CZ-ISCO × sphere only — it is *not* a full region × education × age cube. Region/education/age salary adjustments are designed but not in v1. This is stated explicitly in the README's "Limitations" section.

WHAT THE LOCAL ESCO INDEX GIVES US

A hand-curated CSV of 28 ISCO-08 occupation codes covering all major groups (managers, professionals, technicians, clerical, services, agricultural, trades, drivers). Each row has EN+CZ labels and bilingual keywords for keyword-overlap retrieval. The classify step uses this for top-k retrieval before passing to Claude. Stretch task 29 would extend this to the full ESCO occupation→skill mapping via the ESCO API.

Limitations & honest disclosures

- **No full ISPV multidim cube** — public dataset is CZ-ISCO × wage sphere only. Region/education/age adjustments are designed but not in v1.
- **Eurostat SES is 2022 (quadrennial)** — international anchors will lag.
- **"Personality" renamed** to leadership/ownership/growth signals — CV text cannot defensibly infer personality; CV-observable signals are the honest proxy.
- **No labelled ground truth** — validation is rank-order on a synthetic pack, not absolute correctness.
- **Regex PII redaction** — production would use NER (Named Entity Recognition); regex doesn't catch names or addresses.
- **No institution-prestige scoring** — deliberately excluded to avoid bias.
- **OCR not implemented** — image-only PDFs flow through as low parse confidence; the salary range widens accordingly.
- **v1 ships CZ-only at runtime** — EU/US/UK data layers are designed, not implemented.
- **ISCO classification is hand-curated, not full ESCO** — 28 occupations covering common roles; specialty roles fall back to keyword search and confidence-gated rollup.

> **What this is, what this isn't.**
>
> This is an interview-grade prototype, not a production compensation engine. The methodology is designed to be auditable and honest about its uncertainty (confidence labels, range widening, rollup on low confidence) rather than to be the most precise estimate. A reviewer should be able to follow the math and disagree productively with specific weights — the architecture is correct.

Job Fit & Salary Estimator v0.1.0 · Czech AI-engineering case study deliverable
Spec: `docs/specs/2026-05-04-job-fit-estimator-design.md` · Plan: `docs/plans/2026-05-04-job-fit-estimator.md`
Test report: `docs/reports/test-report.html` · Coverage: `docs/reports/coverage/index.html`