Proposal Paper

Solving the Radar Axis Standardization & Evidence-Weighted Matching Problem

Date: January 07, 2026

Executive Summary

Internities' core product promise is that companies can describe an internship role and obtain a "Company Radar",students can describe themselves and obtain a "Student Skill Radar" and the platform produces a high-accuracy, explainable match score.

The current gap: a radar chart is only meaningful if its axes are comparable. If every company and every student can invent axes freely, the platform risks low comparability, low accuracy and no trustworthy recommendations. If axes are fixed and rigid, the platform risks being too generic to capture what makes a role unique.

This paper proposes a hybrid solution that maximizes both comparability and flexibility:
A set Axis library and custom skill radars for every role

The result is a system that can scale across role types, remain explainable, allow companies to express nuance, and allow students to prove claims—while keeping match scores calibrated and trustworthy.

Contents

1. The Axis Problem and Why It Matters

A radar chart is a visualization of a vector: each axis is a dimension, and the polygon shape communicates relative strengths/requirements. The quality of any match score derived from radars depends on whether the axes mean the same thing across profiles and if the single Axis scores are evidence based.

In practice, internship roles vary hugely. A “Strategy Consulting Intern” and a “Backend Engineering Intern” share very few relevant dimensions. Even within the same role family, companies emphasize different traits (e.g., “client readiness” vs “analytical rigor”). Our product needs enough flexibility to represent this nuance without destroying comparability.

There are several risk scenarios:

- If companies define arbitrary axes, Student Radar and Company Radar may not overlap → matching becomes unreliable or impossible.
- If matching relies only on axis name similarity, it becomes fragile (synonyms, phrasing, language, jargon).
- If matching requires exact axis equality, we force everyone into the same template → roles become generic and users lose trust.
- If weuse AI to infer axes on the fly without governance, we get non-repeatable match results and hard-to-explain behavior.

Therefore the platform needs an explicit strategy for: (a) defining possible axes, (b) mapping and normalizing axes, (c) scoring each axis in a way that is consistent, and (d) combining axis scores with qualitative reasoning and evidence.

2. Design Principles for a Trustworthy Matching System

Comparability: A “7/10 in Python” should mean roughly the same across students and roles.

Flexibility: Companies must be able to express role nuance (especially for top-tier internships).

Explainability: Company Users should understand why a match is high/low at the axis level and overall.

Evidence-first: Claims should be grounded in proofs (projects, grades, assessments, references).

Calibration over aesthetics: The radar is a UI; the underlying model must be statistically stable and monitored.

Fairness and bias control: The system must be tested for prestige bias, demographic proxies, and other distortions.

Human-in-the-loop: AI should assist; students and recruiters remain the decision makers.

3. Proposal Part I — Axis Standardization Without Losing Flexibility

3.1 Recommended approach: Database axis library and different axes per role

- We will be defining a set of (x) axes in our database which is not repetitive, applicable and is sufficient for all possible Industries we are targeting

- After filling out the questionnaire for the role creation, the AI builds a customized skill radar for this specific role. The company will then later be able to change the Axes with an assisting AI Chatbot under the constraint of the defined axis set.

- In the application process, the student has their set of skills and abilities which are then matched on every role individually. Technically, the AI builds a new skill radar for every role the student applies for by applying the skills and abilities the student has to the set role radar and matching both with each other.

3.2 What goes into axis library

Treat axes as a product asset that we curate and version. The library should be:

- Multi-type (hard skills, soft skills, work styles, motivations, constraints).
- Role-family aware (consulting, finance, marketing, software engineering, data science, product, design…).
- Language Customized – for every European market a different set of axes has to be defined
- Versioned (axes definitions can improve over time without breaking historical match scores).

Each axis in the library is not just a name. It must include a structured definition package:

- Axis ID (stable, internal key)
- Name + synonyms (for UI + matching)
- Definition (what it measures, what it does NOT measure)
- Scoring rubric (0–10 with anchors; e.g., 2, 5, 8 examples)
- Evidence types that can support the axis (projects, grades, certificates, tests, recommendations…)
- Assessment options (optional micro-tests that calibrate scoring)
- Typical roles where the axis is used (helps suggest axes during role posting)

3.3 How Companies Create a Company Radar in This Model

A company should never start from a blank radar. Instead, during role posting, the AI proposes a “Role Template” based on role title, function, and description. The recruiter then edits:

- Change the axes, if necessary, with the AI chatbot
- Set importance weights (e.g., 0–5) and/or mark as Must-Have / Nice-to-Have.
- Provide a short rationale per axis (why it matters for success in the role).
- Edits minimum requirements (e.g. GPA of 8) which are evidence-based constraints the student must fulfill to even be matched on the rolls` skill radar

3.4 Axis Mapping: How to match when the student has a new skill radar for every role?

- Axis ID matching: when the student fills out the questionnaire and uploads his or her evidence, the axis score for each possible axis from the defined set is automatically created and evaluated based on the evidence. These Axis scores are checked minimum 2 times by different trained AI models. When applying to a new role, the subset of axes that the company used for their radar is not changed but automatically being used to create the student radar with the same axis subset.
- Prompt-to-complete: for high-weight missing axes (a high weighted company axis that is not weighted on the student side, ask the student to add evidence before applying.
- Hard stop: if a must-have axis cannot be evaluated at all, mark as “Not enough data” and reduce match score (not a silent guess) - this prevents AI hallucinating

4. Proposal Part II — Evidence, Reasoning, and Confidence-Weighted Matching

4.1 Why “Hard Scores Only” Will Not Produce the Matches we Want

A single numeric score per axis is appealing, but it hides critical context:

- Two students can both score 7/10 in ‘Leadership’, but one has led a 20-person society with documented outcomes and the other only self-reports.
- A student’s score may be correct but outdated (skill decay) or not relevant to the role context.
- Some axes are inherently qualitative (‘communication clarity’, ‘structured thinking’) and require narrative evidence.

- Recruiters care about the “why” (how a student demonstrates a trait), not just the “what” (a number).

Therefore, the matching system must incorporate reasoning and proof in a structured way. The solution is to separate: (a) the axis score, (b) the confidence in that score, and (c) the evidence and narrative that justifies it.

4.2 Proposed Axis Scoring Model: Score × Confidence

For each axis, store multiple fields instead of one:

- Score (0–10): estimated capability/fit in that dimension.
- Confidence (0–1): how certain the system is about the score. (verified docs > public portfolio > self-report)
- Reasoning summary (short text): how the score was derived, explicitly referencing evidence.

4.3 Evidence Types and “Proof Upload” Design

To support an exclusive, high-signal marketplace, treat evidence as a first-class object in the product:

- **Academic:** Transcript, grades for relevant courses, honors, scholarships.
- **Portfolio:** GitHub repos, project write-ups, design portfolios, research papers.
- **Work & internships:** Offer letters, contracts, performance reviews, reference contacts.
- **Competitions & awards:** Case competitions, hackathons, Olympiads, prizes.
- **Assessments:** Assessments – later in the Timeline (best option: strategic partnerships with Online assessment companies)
- **Social proof:** Verified recommendations or endorsements (structured, not free-text only).

Verification should be progressive:

- Unverified: user uploaded / linked.
- Auto-verified: checksum/metadata, domain validation, basic fraud detection.
- Institution-verified: e.g., university email, transcript check, or partner verification.
- Company-verified: confirmed by employer during/after internship (powerful feedback signal).

5. Matching Algorithm Blueprint

5.1 Multi-Stage Matching (Recommended)

A robust system should not rely on one single model. Use a pipeline:

- Stage 0 - Hard filters: eligibility (visa/work authorization if relevant), start date, location constraints, language minimum, program year.
- Stage 1 - Must-Have gates: if any must-have constraints < threshold OR axis is ‘unknown’, reduce score sharply or mark as not eligible.
- Stage 2 - Weighted axis score: compute a score using subset axes with confidence and evidence strength.
- Stage 3 - Narrative alignment: compare the student’s axis reasoning + role axis rationale for consistency and relevance.
- Stage 4 - Explainability: generate a shortlist explanation (top reasons, top gaps, and how to improve).

5.2 A Concrete Scoring Formula (Starting Point)

We should implement a first-version scoring formula that is transparent and easy to back test:

For each axis i in the custom created student radar:
• s_i = student score (0–10)
• c_i = confidence (0–1)

Define an effective score for each axis:
eff_i = s_i * c_i

Then compute qualification score assuming there are 10 axes in the role:

$\sum \frac{eff.stu.sco}{comp.sco}$ (where $eff.stu.sco \leq comp.sco$)

This formula will lead to a percentage of matching

5.4 Missing Data Handling

Missing axes is unavoidable. The key is to make missingness explicit:

- Unknown axis score is not the same as low axis score.
- For must-have axes, unknown should behave like “not eligible until assessed”.
- For nice-to-have axes, unknown should reduce score slightly and trigger a prompt to add evidence.
- Give students fast ways to fill gaps: micro-assessments, quick evidence upload, or structured Q&A.

6. Governance, Fairness, and Quality Control

6.1 Axis Governance

Because axes define the language of our marketplace, we need governance from day 1:

- Axis owner: internal responsibility (product/ML) for the library.
- Axis review cadence: monthly review of “library candidate” axes coming from companies.
- Axis versioning: never change definitions silently; increment versions and maintain mapping for historical comparisons.
- Deprecation policy: old axes can be hidden from UI but kept for historical match reproducibility.

6.3 Anti-Gaming Measures

As soon as scoring affects opportunities, users will optimize it. Design for this:

- Separate “self-reported” vs “verified/assessed” contributions to axis scores.
- Use confidence/evidence strength to down-weight unsupported claims.
- Detect inconsistencies (e.g., claiming advanced Python but no evidence + failing micro-assessment).
- Rate-limit sudden large score changes unless new evidence is added.
- Keep rubric anchors visible (students know what 8/10 means, making the system feel fair).

7. Implementation Roadmap (MVP → Production)

7.1 MVP (Fast, Reliable, Explainable)

For our MVP, prioritize stability and trust over model sophistication:

- Start with 40–80 canonical axes across 6–8 role families.
- Role posting uses templates + limited axis customization.
- Student onboarding generates initial axis scores using structured Q&A + CV parsing (optional).
- Evidence objects + verification states exist from day 1 (even if verification is manual early).
- Matching uses the transparent formula (Score × Confidence)
- Track feedback signals: apply → interview → offer → performance (where available).

7.2 Post-MVP (Data-Driven Calibration and Model Improvement)

After pilots, you can improve accuracy via learning loops:

- Calibrate axis scoring with outcomes data (which scores predict interviews/offers).
- Refine weights per role family using historical hiring success patterns.
- Expand axis library based on repeated extension requests.
- Introduce assessments for high-stakes axes (coding, finance modeling, writing).
- Build dashboards for companies showing how Internities reduced screening time and improved interview-to-offer conversion.