Semantic AI-Based Recruitment and Skill Gap Analysis System

A Social-AI Talent Hub

Prepared by:

1. Haba Mohammad Hamed
2. Heba Yaser Mohammad
3. Abdullah Mahmoud Abdullatif
4. Mohammad Ahmad Sayed
5. Osama Ahmad Elneny

Date: February 24, 2026

1. Executive Summary

This proposal presents a production-ready design for a Social-AI Talent Hub that blends a professional network with semantic, explainable recruitment and personalized skill-gap coaching. The platform delivers three unified experiences: (a) an AI-aware social feed where engineers publish posts and projects; (b) Smart Matching that computes job fit from a live, continuously enriched profile—without uploading a CV each time; and (c) an HR Dashboard that surfaces the most relevant and active candidates with transparent, data-driven scores. The result is a private talent ecosystem that helps companies hire faster and helps engineers upskill with precise guidance.

2. Problem Statement and Goals

Hiring top technical talent is noisy and slow: static resumes go out-of-date, CV screening is manual, and the best candidates are hard to spot among large applicant pools. Engineers also struggle to see exactly what to improve to reach specific roles.

- Replace manual CV screening with live, validated evidence from posts, projects, and skills.
- Provide instant, explainable job–candidate matching without repeated file uploads.
- Deliver personalized learning guidance to close skill gaps for targeted roles.
- Give HR a single decision surface with activity signals, match scores, and messaging.

3. Solution Overview: Social-AI Talent Hub

The platform comprises three primary modules (Tabs):

1. Feed (Home): Engineers share posts and projects. AI extracts skills, frameworks, and topics from content and updates profile evidence and experience points.
2. Smart Matching (Jobs): For any job card, the system computes an on-the-fly fit score from the candidate's profile, projects, and social evidence.
3. HR Dashboard: Recruiters see active, high-fit candidates, explainable score breakdowns, and can message candidates directly.

4. Key Features by Persona

4.1 Engineers

- Smart Profile: unified skills, projects, certifications, and auto-inferred strengths.
- Evidence Graph: posts, code snippets, repos, badges, and endorsements increase skill confidence.
- Instant Job Fit: per-role fit with strengths and areas to improve.
- Career Coach: prioritized learning plan to move from current fit to target fit.

4.2 HR/Recruiters

- Ranked Talent Lists: top candidates for each role with real-time fit scores.
- Explainability: transparent breakdown of why someone is a fit (skills, projects, activity).
- Direct Outreach: one-click messaging and saved talent pools.
- Private Ecosystem: engage a community pre-interested in the brand and domain.

4.3 Admin

- Role-Based Access Control (RBAC) and audit logs.
- Taxonomy management for skills/keywords and domain ontologies.
- Content moderation and abuse detection.

5. AI and Semantics

The AI layer combines named-entity recognition for skills and tools, embedding-based semantic similarity for job–profile matching, and rule+model hybrids for explainability. A domain ontology (e.g., Automotive/Embedded, Cloud, Data) steers inference and recommendations.

- Skill Extraction: parse posts/projects to detect languages, frameworks, libraries, domains, and seniority cues.
- Profile Inference: aggregate multi-source evidence with confidence scores and decay over time.
- Job-to-Skill Mapping: decompose job descriptions into normalized skills with weights.
- Semantic Matching: compute fit using weighted cosine similarity between job and profile vectors plus rule-based bonuses.
- Explainability: attach each score component to concrete evidence snippets (post, project, certification).
- Skill-Gap Coaching: identify missing or weak skills and propose an ordered, time-bounded learning path.

6. Data Model (High-Level)

Core entities and suggested tables:

- Users, Roles, Permissions
- Profiles (headline, bio, location, seniority)
- Skills (normalized), UserSkills (level, evidence, confidence, last_seen)
- Experience, Education, Certifications
- Posts, Comments, Likes, Projects, Attachments
- Companies, Jobs, JobRequirements (skill_id, weight), Applications
- MatchScores (per user–job), Messages, Notifications

7. Matching and Skill Gap Scoring

Let J be the job vector and P the profile vector (skills weighted by confidence). The overall fit combines semantic similarity with evidence-based bonuses and activity signals:

```
Fit = w_s * cosine(J, P)
      + w_e * EvidenceBonus(P)
      + w_a * ActivitySignal(User)
      + w_x * ExperienceAlignment(P, J)
Weights default: w_s=0.55, w_e=0.20, w_a=0.10, w_x=0.15.
```

Skill Gap = sorted list of (required_skill, required_level − current_level, impact=job_weight) with recommendations mapped from a curated learning catalog.

8. System Architecture

- .NET 8 (ASP.NET Core) microservices: Identity, Profiles, Social, Jobs, Matching, Messaging, Admin.
- Data: SQL (transactional), Redis (cache), Object Storage for media, optional Vector Store for embeddings.
- Search & Analytics: full-text search engine, event streaming for activity signals.
- AI Layer: embedding models, NER, lightweight classifiers; offline trainers + online inference endpoints.
- API Gateway + Auth (OAuth2/OIDC), RBAC, rate limiting.
- Observability: centralized logging, metrics, tracing, and alerting.

9. UI/UX Highlights

- Left sidebar for navigation: Home, Jobs, Messages, Profile, Admin.
- Job cards with live Fit Score and expandable explanation (strengths & improvements).
- HR dashboard with ranked candidates, filters, and message composer.
- Accessible design, keyboard shortcuts, and responsive layout.

10. Security, Privacy, and Ethics

- Privacy by design: explicit consent for AI processing and public visibility controls.
- Data minimization: store only necessary data; allow export/delete of profile data.
- Secure by default: TLS, hashed passwords, secrets management, and regular pentests.
- Fairness: monitor score distributions; enable bias audits and human-in-the-loop overrides.

11. Non-Functional Requirements

- Scalability: autoscaling microservices and horizontal read replicas.
- Performance: p95 job-fit computation < 300 ms; feed ingestion < 1 s.

- Availability: 99.5%+ for MVP, 99.9% for v1.0.
- Reliability: idempotent writes and exactly-once processing for critical events.

12. DevOps and CI/CD

- Branching strategy with automated builds, tests, security scans, and container images.
- Blue/green or rolling deployments; feature flags for gradual rollouts.
- Backup/restore playbooks and disaster recovery RTO/RPO objectives.

13. Delivery Plan (MVP → Beta → v1.0)

Illustrative 12–14 week plan:

4. Weeks 1–2: Architecture, data model, UI shell, identity and RBAC.
5. Weeks 3–4: Profiles, Posts/Comments/Likes, evidence ingestion, basic search.
6. Weeks 5–6: Jobs, job requirements editor, baseline matching service.
7. Weeks 7–8: HR dashboard, messaging, notifications, explainability views.
8. Weeks 9–10: Skill-gap coaching, learning catalog integration, analytics.
9. Weeks 11–12: Hardening, security review, performance tuning, beta.
10. Weeks 13–14: Feedback, polish, and v1.0 release.

14. Key Metrics (KPIs)

- Time-to-shortlist per role (days).
- Offer acceptance rate and time-to-hire.
- % of candidates improving fit after coaching (delta fit).
- HR dashboard adoption and message response rate.

15. Risks and Mitigations

- Cold-start for new users → guided onboarding and starter projects.
- Noisy posts → confidence-weighted evidence with decay and moderation.
- Model drift → scheduled evaluation sets and retraining triggers.
- Privacy concerns → transparent consent flows and data export/delete.

16. Future Extensions

- Graph-based recommendations (teams, mentors, learning buddies).
- Company-branded communities and events.
- Proctoring for skills assessments and verified badges.
- Multi-tenant SaaS with usage-based billing.

Appendix A: Sample API Endpoints

```
POST /api/posts -> create a post
GET  /api/jobs -> list jobs with live fit
POST /api/match/score -> compute explanation for a user-job pair
GET  /api/hr/candidates?jobId=... -> ranked candidates with filters
```

Appendix B: Sample UI Mockups

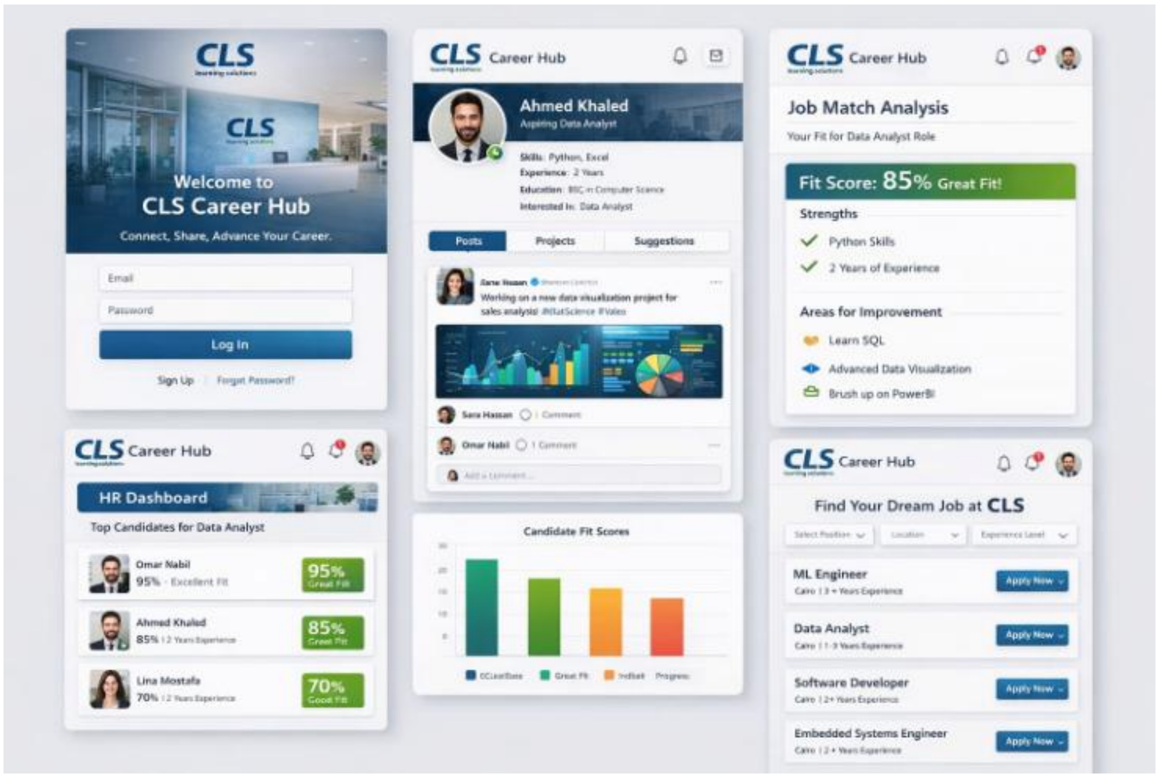

Figure 1. Career Hub UI Mockup