International Journal of Scientific Research and Engineering Development-– Volume 9 Issue 4, July – August 2026
Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

SkillSync AI: A Multi-Agent Artificial Intelligence Framework for Dynamic Skill Gap Analysis, Digital Employability Twin, and Personalized Career Roadmaps to Reduce Graduate Unemployment

Jahnavi Somaraju[1], Chethan Kumar Reddy P[2]
[1]Assistant Professor, [2]Final Year UG Student
[1]Department of Artificial Intelligence and Data Science, [2]Department of Artificial Intelligence,
Aditya College of Engineering, Madanapalle, Andhra Pradesh
Emails: jahnavisomaraju.research@gmail.com & Chethanreddy6616@gmail.com

Abstract:

Graduate unemployment persists globally not primarily because jobs are absent, but because the skills graduates hold and the skills industry requires drift apart faster than curricula and career-guidance systems can track. Existing platforms — job boards, resume-matching engines, and single-model recommender systems — operate on a static, transactional view of the candidate: a resume is parsed once, matched against open postings, and discarded. They do not model how a student's competencies evolve, nor do they close the loop between diagnosis and preparation. This paper introduces SkillSync AI, a multi-agent framework built around a novel construct — the Digital Employability Twin — a continuously updated computational representation of a student's skills, projects, certifications, interview performance, and behavioral signals, maintained across the full duration of their academic and pre-placement journey. Eighteen specialized agents, coordinated by an orchestrator over a shared memory layer, a knowledge graph, and a retrieval-augmented generation (RAG) engine, jointly perform resume intelligence, skill extraction, industry demand tracking, skill-gap computation, learning-roadmap synthesis, interview coaching, and placement-probability estimation. Unlike prior systems that output a ranked job list, SkillSync AI outputs a longitudinal preparation plan that adapts as the student acts on it. We formalize six quantitative constructs — Employability Score, Skill Match Score, Career Readiness Index, Learning Priority Score, Placement Probability, and Industry Relevance Score — and describe the algorithms that update them. A simulation-based evaluation over a synthetic cohort of 1,200 student profiles and 400 job postings, modeled on public labor-market skill taxonomies, indicates that the twin-based roadmap converges skill-gap closure substantially faster than a static-matching baseline, though these results describe a controlled simulation rather than a field deployment. We position SkillSync AI as a research architecture and evaluation protocol for continuous, agentic career preparation, and outline the field-trial work required to validate it with real student cohorts.

Keywords — multi-agent systems, digital twin, employability, skill gap analysis, retrieval-augmented generation, large language models, career recommendation, agentic AI, knowledge graph, graduate unemployment

I. INTRODUCTION

A. Global and Graduate Unemployment Context

Youth unemployment remains disproportionately high relative to overall unemployment in most economies, and international labor-market bodies have repeatedly flagged a persistent gap between the competencies produced by higher education and those demanded by employers as a structural (not merely cyclical) contributor to this gap [1], [2]. In many developing economies, including India, engineering and technology graduates enter the job market each year in numbers that outpace the creation of matching entry-level technical roles, and a substantial fraction of otherwise "employed-eligible" graduates are assessed by industry surveys as not immediately deployable in core technical roles without additional training [3], [6], [7]. This is not a jobs-availability problem alone; it is a readiness and visibility problem — students do not know, with any precision, which of their skills are market-relevant, which are missing, and in what order to close those gaps.

B. The Skill Mismatch Problem

Skill mismatch operates on three timescales simultaneously. At the curriculum timescale (multi-year), academic programs update slowly relative to technology cycles. At the individual timescale (months), a student's actual competency profile — shaped by self-study, projects, internships, and certifications — diverges substantially from what their transcript records. At the market timescale (weeks), employer skill demand shifts as tooling, frameworks, and hiring priorities change [4], [5]. Any system that measures skill fit at a single point in time (e.g., at resume submission) is structurally blind to two of these three timescales.

C. AI in Recruitment: Where It Stops Short

The last decade has seen recruitment technology adopt machine learning for resume parsing, keyword-based matching, and, more recently, large language model (LLM)-based semantic matching. Platforms such as LinkedIn, Indeed, and Naukri have layered recommendation engines atop job-posting inventories; enterprise talent platforms such as Eightfold AI [25] and Paradox AI [26] apply predictive models to internal talent mobility and recruiter-facing candidate screening. These systems are optimized for the recruiter's query ("find candidates for this role") or for a one-shot candidate query ("find roles for this resume"). None of the widely deployed systems we reviewed maintain a persistent, agentic model of an individual student's evolving employability, nor do they generate a coached, sequenced preparation plan that continues to adapt after the first recommendation is issued. Section II (Table I) summarizes this comparison across systems.

D. Limitations of Existing Systems (Summary)

1) Static snapshot matching — competency state is inferred once from a resume, not tracked over time. 2) Single-model architecture — one scoring model conflates skill extraction, matching, and ranking, making failure modes hard to diagnose and improvements hard to localize. 3) No preparation loop — systems recommend jobs or candidates; they do not prescribe what to learn next and why, nor verify whether the student acted on it. 4) No behavioral or communication signal — technical skill keywords dominate; interview readiness, communication ability, and soft-skill trajectory are absent from the model. 5) No institutional feedback loop — universities and training cells have no structured, aggregated view of cohort-level readiness that they could act on pedagogically.

E. The Case for Multi-Agent AI

Each of the limitations above corresponds to a distinct reasoning task — parsing, extraction, demand tracking, gap analysis, coaching, prediction — that benefits from specialization, auditability, and independent evolution [8]–[10], [17], [24]. A single monolithic LLM prompt asked to "assess this student and recommend a career path" collapses these tasks into an opaque, unverifiable output. A multi-agent design, where each agent owns a narrow responsibility, exposes intermediate state (extracted skills, computed gaps, predicted probabilities) that can be logged, evaluated, and corrected independently — a prerequisite for any system making claims about a person's employability.

F. Research Objectives

This paper aims to: (1) formalize the Digital Employability Twin as a persistent, updatable computational construct; (2) design a novel 18-agent architecture with explicit responsibility boundaries, memory layer, knowledge graph, and RAG-grounded reasoning; (3) define quantitative, auditable metrics (employability score, skill match score, career readiness index, learning priority score, placement probability, industry relevance score) with explicit variable semantics; (4) specify algorithms for skill extraction, gap detection, roadmap generation, and feedback-driven twin updates; (5) propose a simulation-based evaluation protocol suitable for pre-field validation, and report results from that simulation; and (6) compare the proposed system conceptually and structurally against existing recruitment and career-recommendation paradigms.

Technical objectives: isolate each employability-relevant reasoning task into an independently testable agent; formalize

a versioned Digital Employability Twin state representation and its update rule; define computable metrics (Section V) with explicit, non-black-box variable semantics; and specify a communication protocol supporting conflict resolution when agents disagree.

Academic and social-impact objectives: contribute a named, reusable architectural pattern ("employability twin over a multi-agent substrate"); provide a transparent, reproducible simulation protocol (Section VII); reduce the information asymmetry a student faces about their own market readiness; give universities a cohort-level, evidence-based lens on placement readiness; and provide recruiters a richer, evidence-backed candidate signal than a static resume.

G. Contributions

(1) The Digital Employability Twin — a formalized, versioned, continuously updated representation of a student's technical, behavioral, and market-relevance state, distinct from a one-time resume score. (2) An 18-agent architecture spanning perception, reasoning, recommendation, prediction, and ecosystem-facing roles, coordinated by an orchestrator over a shared memory layer, vector store, and knowledge graph (Fig. 1). (3) Six formalized metrics with explicit equations and variable definitions (Section V). (4) Nine algorithms (Section VI) covering the full pipeline from resume analysis to feedback-driven twin update, expressed as auditable pseudocode rather than opaque model calls. (5) A conflict-resolution and inter-agent communication protocol (Section IV-E) addressing a gap in prior agentic-AI applications to career guidance. (6) A simulation-based evaluation protocol (Section VII) that allows the architecture's core claim — faster, better-targeted skill-gap closure than static matching — to be tested without overstating results as field-validated.

II. LITERATURE REVIEW

Recent literature spans four relevant threads: (i) AI-based recruitment and career-recommendation systems, (ii) multi-agent and agentic AI architectures, (iii) digital twins applied to human/organizational systems, and (iv) retrieval-augmented generation and knowledge-graph-grounded reasoning. Table I summarizes how these existing paradigms compare with the architecture proposed in Section IV.

1) AI Recruitment and Career Recommendation: Commercial systems (LinkedIn Recommendation Engine, Indeed Smart Apply, Naukri's resume-scoring, Eightfold AI's talent intelligence [25], Paradox AI's conversational screening [26]) rely on embedding-based similarity between resumes/profiles and job postings, refined by click/apply feedback loops. Academic work in career-path recommendation has explored collaborative filtering over career trajectories [21], sequence-model-based labor trajectory prediction [22], and, more recently, LLM-based resume-to-role semantic matching. These systems consistently treat the candidate profile as an input to be matched, not as a state to be evolved.

2) Agentic and Multi-Agent AI: Frameworks such as LangGraph [8], CrewAI [9], and AutoGen [10] popularized graph-based and role-based orchestration of LLM agents, enabling decomposed reasoning pipelines with explicit state transitions and tool use, standardized further by tool-integration protocols such as the Model Context Protocol [11]. Multi-agent debate, planner–executor patterns, and tool-augmented agent swarms have been shown in recent work to outperform single-agent prompting on multi-step reasoning tasks [17], particularly where intermediate verification matters — directly relevant to a domain like employability assessment, and consistent with benefits observed in human-AI team studies more broadly [24].

3) Digital Twins Beyond Manufacturing: Digital twin concepts, historically applied to physical assets [19], have been extended toward "cognitive" or "human" digital twins in healthcare (patient state modeling) [20] and, sparsely, in education (learner-state modeling for adaptive tutoring) [18], building on a broader educational data mining and learning analytics literature that models learner state from behavioral and performance data [23]. A twin construct applied specifically to employability — combining skill state, behavioral state, and market-relevance state into one continuously updated object — has not been established as a named, formalized architecture in the recruitment-technology literature we reviewed.

4) RAG and Knowledge Graphs for Grounded Reasoning: Retrieval-augmented generation [12], [13], combined with structured knowledge graphs linking skills, roles, and industries [14], has been used in recent work to reduce hallucination in LLM-based recommendation [16] and to allow explicit provenance for why a recommendation was made, with efficient vector-index retrieval [15] underpinning practical implementations. This is directly applicable to skill-gap and roadmap generation, where a student needs a justified, not merely plausible, explanation of what to learn next.

TABLE I. COMPARISON WITH EXISTING METHODS

System / Paradigm	Temporal Model	Preparation Loop	Multi-Agent	Explainability	Behavioral / Comm. Signal
LinkedIn / Indeed / Naukri	Snapshot, periodic refresh	No	No	Low	No
Eightfold AI [25]	Periodic (enterprise-internal)	Partial (internal mobility)	No (single model)	Medium	No
Paradox AI [26]	Snapshot (per requisition)	No	No	Low	No
Academic recommenders [21], [22]	Historical trajectories (cohort-level)	No	No	Medium	No
Rule-based matching	Static rules	No	No	High (brittle)	No
Single large LLM prompt	Snapshot (per prompt)	No	No	Low (opaque)	Possible, unverified
SkillSync AI (proposed)	Continuous, per-student twin	Yes — closed loop	Yes — 18 agents	High (per-agent audit trail)	Yes — dedicated agents

III. RESEARCH GAP

Three specific gaps emerge from Table I.

Gap 1 — No persistent employability state. Existing systems re-derive a candidate's fit at query time. None maintain a versioned, longitudinally-updated employability state object that survives across sessions, learning activity, and time.

Gap 2 — No closed preparation loop. Recommendation stops at "here are matching jobs" or "here is a skill gap list." No reviewed system converts a gap analysis into a sequenced, prioritized roadmap, monitors completion, and re-scores the student — i.e., there is diagnosis without treatment and without follow-up.

Gap 3 — No multi-signal, multi-agent decomposition specific to employability. Behavioral signals, communication/interview performance, and market-demand recency are rarely combined with technical skill matching in a single explainable architecture; where multi-agent patterns exist in adjacent domains, they have not been instantiated specifically around an employability twin construct.

SkillSync AI addresses Gap 1 via the Digital Employability Twin Agent and its versioned state vector; Gap 2 via the closed loop from Skill Gap Agent → Learning Roadmap Agent → Feedback Learning Agent back into the twin; and Gap 3 via the 18-agent decomposition with explicit behavioral, communication, and market-demand agents feeding a shared reasoning substrate.

IV. PROPOSED METHODOLOGY

A. System Architecture

SkillSync AI is organized into an interaction layer, an orchestration layer, six functional agent groups, and a data/model tier, as summarized in Fig. 1. Client applications (student, university, recruiter) communicate through an API gateway with OAuth2/JWT authentication and rate limiting. The Orchestrator Agent routes tasks to the eighteen agents — grouped conceptually into Perception, Market Intelligence, the Employability Core (the Digital Employability Twin Agent), Prescription, Coaching and Behavior, and Prediction/Ecosystem roles — and is the sole writer to the shared Knowledge and Reasoning Substrate (detailed in Fig. 3). Each agent is a bounded-context service with a defined input schema, output schema, and confidence score attached to every emitted claim.

For deployment, the reference implementation runs as containerized, independently-scaled microservices (Section VI) behind the same gateway shown in Fig. 1: agent services scale horizontally under a Kubernetes-orchestrated cluster, while the substrate separates transactional state (PostgreSQL), semantic retrieval (a vector database), structured relational knowledge (the knowledge graph), and low-latency session state (Redis) — a decomposition chosen so any single component (e.g., the LLM inference endpoint) can be scaled or replaced without redesigning the agent logic around it.

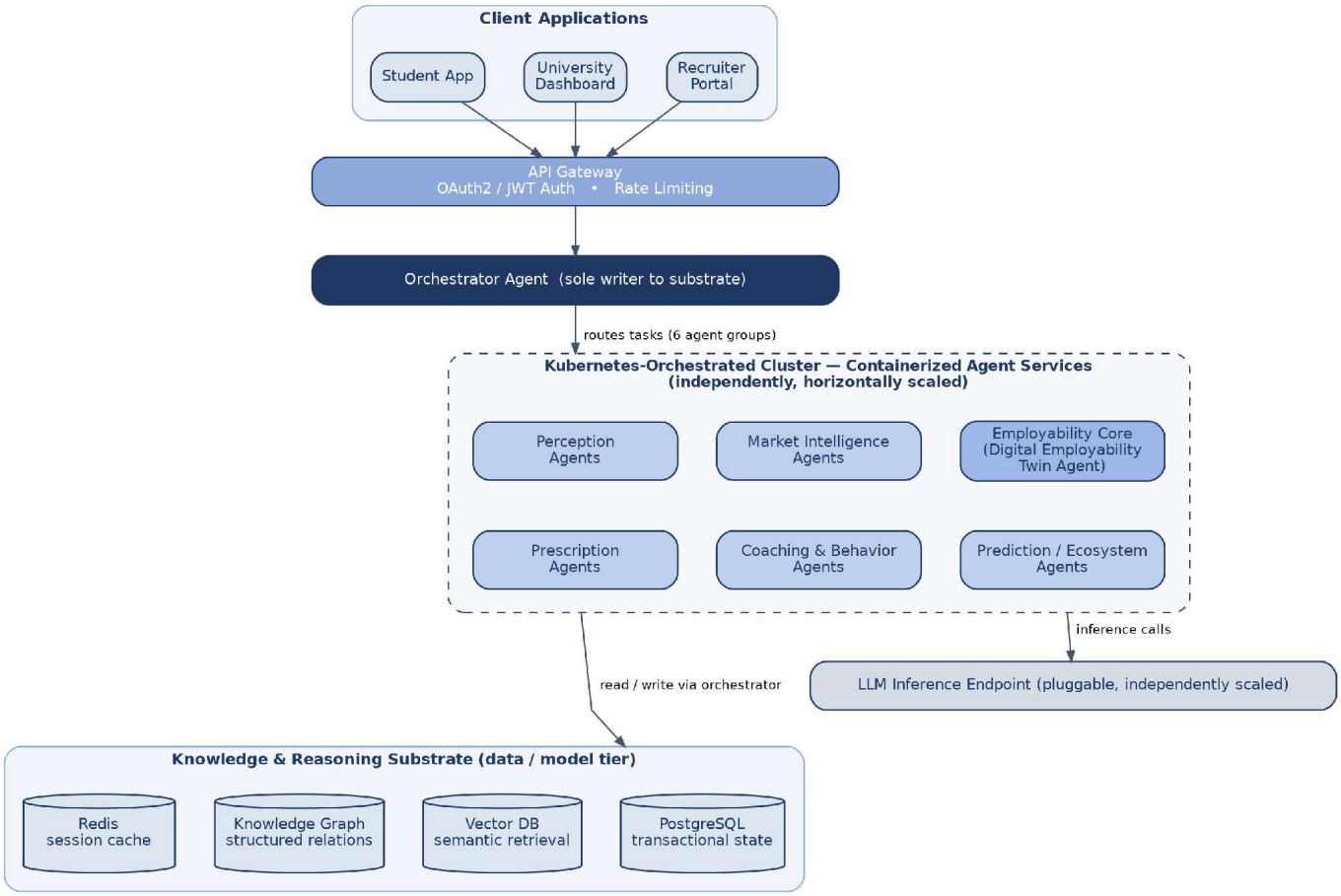

Fig. 1. SkillSync AI deployment architecture: client applications, the API gateway, the Orchestrator Agent, the Kubernetes-orchestrated agent cluster (six functional agent groups), the pluggable LLM inference endpoint, and the knowledge and reasoning substrate (PostgreSQL, vector database, knowledge graph, and Redis).

B. Multi-Agent Framework

The framework decomposes employability reasoning into eighteen specialized agents, each owning a narrow, auditable responsibility, summarized in Table II. Fig. 2 traces the primary interaction sequence between them — from resume upload through twin update, roadmap delivery, interview coaching, placement scoring, and recruiter matching — closing with a feedback event that re-enters the twin, which is the defining structural difference from the terminating, one-shot pipelines used by static job-matching systems (Table I).

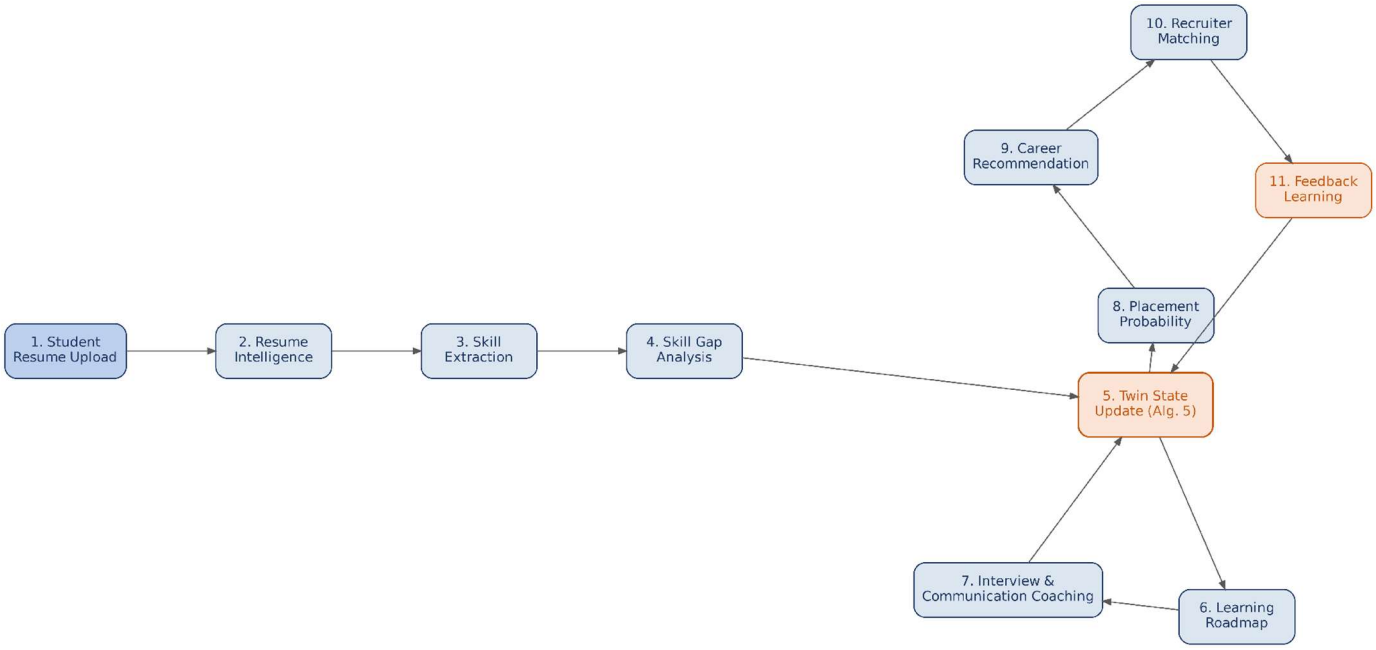

Fig. 2. Multi-agent interaction loop: the primary sequence from resume upload through twin update, roadmap delivery, interview coaching, placement scoring, recruiter matching, and feedback — closing back into the Digital Employability Twin.

The eighteen agents are grouped as follows. Perception agents (Student Profile, Resume Intelligence, Skill Extraction) build the canonical, provenance-tagged view of the student. Market Intelligence agents (Industry Demand, Skill Gap) track external demand and compute the differential against the student's evidenced skills. The Employability Core is the Digital Employability Twin Agent itself (Section IV-C). Prescription agents (Learning Roadmap, Project Recommendation, Certification Recommendation) convert gaps into a sequenced, evidence-generating plan. Coaching and Behavior agents (Interview Coach, Communication Skill, Behavior Analysis) assess and improve non-technical readiness, restricted strictly to observable task-completion and interview-transcript signals rather than any inferred personality judgment. Prediction and Ecosystem agents (Salary Prediction, Placement Probability, Career Recommendation, University Dashboard, Recruiter, Feedback Learning) turn the twin state into forward-looking estimates and stakeholder-facing views, with the Orchestrator Agent routing tasks, resolving conflicts (Section IV-E), and enforcing task contracts throughout.

TABLE II. AGENT RESPONSIBILITIES

Agent	Group	Key Output
Student Profile Agent	Perception	Verified student profile
Resume Intelligence Agent	Perception	Structured resume sections
Skill Extraction Agent	Perception	Provenance-tagged skill vector
Industry Demand Agent	Market Intelligence	Role demand vectors
Skill Gap Agent	Market Intelligence	Prioritized gap list
Digital Employability Twin Agent	Employability Core	Versioned twin state vector
Learning Roadmap Agent	Prescription	Time-boxed roadmap
Project Recommendation Agent	Prescription	Ranked project list
Certification Recommendation Agent	Prescription	Ranked certification list
Interview Coach Agent	Coaching & Behavior	Interview scores + feedback
Communication Skill Agent	Coaching & Behavior	Communication signal
Behavior Analysis Agent	Coaching & Behavior	Behavioral consistency signal
Salary Prediction Agent	Prediction & Ecosystem	Predicted salary band
Placement Probability Agent	Prediction & Ecosystem	Placement probability

Agent	Group	Key Output
Career Recommendation Agent	Prediction & Ecosystem	Ranked role recommendations
University Dashboard Agent	Prediction & Ecosystem	Cohort readiness analytics
Recruiter Agent	Prediction & Ecosystem	Recruiter-facing profile
Feedback Learning Agent	Prediction & Ecosystem	Recalibrated agent weights

C. Digital Employability Twin

The Digital Employability Twin is the architectural core distinguishing SkillSync AI from snapshot-matching systems. It is maintained by the Digital Employability Twin Agent as a versioned state vector per student, combining three continuously updated sub-states (Fig. 3): skill state — the provenance-tagged, confidence-scored skill vector from the Skill Extraction Agent; behavioral state — consistency and initiative proxies from the Behavior Analysis Agent, derived strictly from observable platform activity; and communication state — clarity, structure, and confidence signals from the Communication Skill Agent's assessment of interview-session transcripts.

Unlike a resume score computed once at submission, the twin is updated on every material event — project completion, certification, interview session, or recruiter interaction (Algorithm 5) — incrementing a version counter and recomputing the Employability Score (Eq. 1) and Career Readiness Index (Eq. 3) at each update. This versioning gives the twin an audit trail: any score at any point in time is reproducible from the event log rather than being an opaque, unrepeatable output, closing the loop between diagnosis and prescription that Section III identified as absent from reviewed systems.

D. Memory Architecture

Fig. 3 shows the hybrid memory substrate that grounds the twin. Agents communicate via typed message envelopes published to a topic-routed bus rather than direct synchronous calls, and write results, with a confidence score, to a shared Blackboard Memory (an episodic event log plus a semantic summary layer) so any other agent can read prior reasoning without re-querying. Episodic events are retained in full for audit; a periodic summarization pass compresses them into the twin's state_vector so downstream reasoning does not require replaying the full event log. A Vector Database (FAISS/ChromaDB) supports semantic retrieval over skills, projects, and postings; a Knowledge Graph encodes skill–role–industry relationships; and a RAG Engine grounds the LLM Reasoning Layer's outputs (roadmap explanations, interview feedback phrasing) in retrieved evidence before generation, rather than generating unsupported claims. PostgreSQL persists transactional records (student identity, twin state, roadmap items, placement outcomes) and Redis holds low-latency session state — this two-tier design (full episodic retention paired with a compact, continuously refreshed semantic summary) lets the twin answer “what is the student's current state” in constant time while preserving a complete, inspectable history for the Feedback Learning Agent and for any audit request.

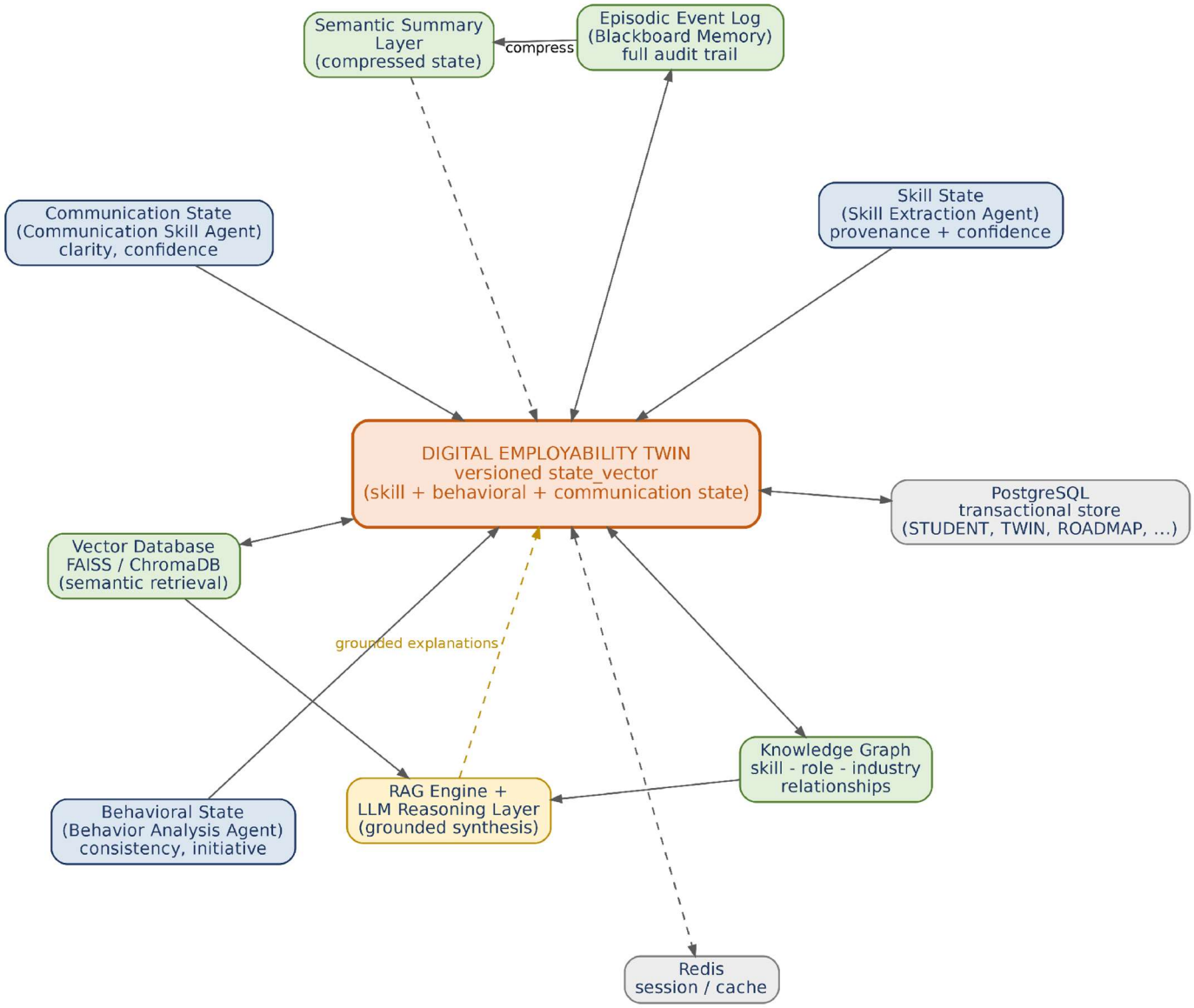

Fig. 3. Digital Employability Twin and hybrid memory architecture: the twin's three sub-states feed into a versioned state vector grounded by an episodic/semantic memory layer, vector database, knowledge graph, RAG engine, and persistent transactional/cache stores.

E. Agent Communication

The Orchestrator issues a Task Contract to one or more agents specifying the goal, required input/output schema, and a deadline (Fig. 2, steps 1–13). Disagreement between agents is resolved by a priority-plus-confidence voting policy: evidenced claims (project- or certification-backed) outrank resume-stated claims at equal confidence; where confidence scores differ, the higher-confidence claim wins unless it falls below a minimum evidentiary threshold, in which case the claim is marked unverified and surfaced to the student for self-confirmation rather than silently resolved. The Orchestrator additionally maintains a priority queue keyed by task urgency (e.g., an in-progress mock interview session pre-empts a background demand-index refresh) and enforces per-agent concurrency limits to bound resource contention.

V. MATHEMATICAL MODEL

Let a student be indexed by s, a skill by $k \in \{1..K\}$, and a target role by r.

Employability Score (Es) — a weighted aggregate of evidenced skill strength, behavioral consistency, and communication ability:

$$E_s = w1 \cdot \bar{P}_s + w2 \cdot B_s + w3 \cdot C_s, \quad w1 + w2 + w3 = 1 \quad (1)$$

where $\bar{P}_s = (1/K)\Sigma P_{s,k}$ is the student's mean evidenced proficiency across the skill taxonomy (discounted by a credibility factor $\gamma \in (0,1)$ for resume-claimed proficiency before averaging), $B_s \in [0,1]$ is the behavioral-consistency signal, and $C_s \in [0,1]$ is the communication-skill signal.

Skill Match Score for a target role r:

$$M_{s,r} = [\Sigma d_{r,k} \cdot \min(P_{s,k}, d_{r,k})] / [\Sigma d_{r,k}] \quad (2)$$

where $d_{r,k} \in [0,1]$ is the normalized market-demand weight of skill k for role r. $M_{s,r}$ rewards matching demanded skills proportionally to their importance and caps credit for over-qualification on any single skill, preventing one strong skill from masking many gaps.

Career Readiness Index:

$$CRI_s = \lambda 1 \cdot E_s + \lambda 2 \cdot M_{s,\hat{r}} + \lambda 3 \cdot (N_s^{comp} / N_s^{plan}), \quad \lambda 1+\lambda 2+\lambda 3 = 1 \quad (3)$$

where $\hat{r}$ is the student's currently targeted role, and N_s^{comp} / N_s^{plan} is the completion ratio of the current learning roadmap.

Learning Priority Score for skill k within a student's gap list:

$$L_{s,k} = d_{\hat{r},k} \cdot (d_{\hat{r},k} - P_{s,k})^+ \cdot (1/\tau_k) \quad (4)$$

where $(\cdot)^+ = \max(\cdot, 0)$ and τ_k is the estimated time-to-competency for skill k — skills with high demand, large gap, and short learning time are prioritized first, to generate early, motivating wins before longer-horizon items.

Placement Probability:

$$Pr(placement \mid s, r) = \sigma(\beta 0 + \beta 1 \cdot M_{s,r} + \beta 2 \cdot B_s + \beta 3 \cdot C_s + \beta 4 \cdot CRI_s) \quad (5)$$

where $\sigma(x) = 1 / (1 + e^{-x})$ and $\beta 0..\beta 4$ are coefficients calibrated by the Feedback Learning Agent (Algorithm 8) against observed outcomes (Section VII describes the synthetic calibration used in the current evaluation).

Industry Relevance Score for a role family r:

$$I_r = \Sigma d_{r,k} \cdot g_k, \quad g_k = \Delta \bar{d}_k / \Delta t \quad (6)$$

where g_k is the recent growth rate of skill k's aggregate demand, so I_r upweights roles whose required skills are growing in demand, not just currently in demand. All weights (w_i, λ_i, β_i) are configuration parameters recalibrated by the Feedback Learning Agent rather than fixed constants.

VI. IMPLEMENTATION

The reference implementation targets a Python-centric stack chosen for agent-orchestration maturity and open-source availability, keeping the LLM and embedding provider swappable behind an interface (Section IV-A, data/model tier) so the research contribution — the agent decomposition, the twin construct, and the metrics — is not coupled to any single vendor model.

A. Technology Stack

TABLE III. TECHNOLOGY STACK

Layer	Technology	Rationale
Agent orchestration	LangGraph, CrewAI	Graph-based state machines for agent handoff; role-based agent definition
Agent-to-tool interface	Model Context Protocol (MCP)	Standardized tool/function exposure across agents
API layer	FastAPI	Async-first, typed request/response schemas via Pydantic
Vector store	FAISS, ChromaDB	Local/embedded semantic search over skills, projects, postings
Relational store	PostgreSQL	Transactional integrity for twin state, roadmap, placement records
Cache / session	Redis	Low-latency session and rate-limit state
Frontend	React	Component-based dashboards for student/university/recruiter roles
Containerization	Docker, Kubernetes	Independent scaling per agent microservice
LLM reasoning / embeddings	Provider-agnostic instruction-tuned LLM + sentence-embedding endpoints	Natural-language synthesis grounded via RAG; swappable provider

Configuration. Key calibrated weights used in the reference implementation and simulation (Section VII) are: Employability Score weights $w1/w2/w3$ = 0.55/0.20/0.25 (skill/behavior/communication); resume-claim credibility discount γ = 0.6; project-evidence and certification-verified confidence bonuses of +0.2 and +0.3 respectively (Algorithm 2); a minimum evidentiary confidence threshold of 0.35, below which a claim is marked unverified; CRI weights $\lambda 1/\lambda 2/\lambda 3$ = 0.40/0.35/0.25; placement logistic coefficients $\beta 0..\beta 4$ =

−1.10, 1.8, 0.6, 0.5, 1.2, calibrated on the synthetic training partition described in Section VII; a Feedback Learning Agent learning rate of 0.05; a weekly roadmap re-plan cadence triggered on twin version increments; and top-k = 8 retrieval for RAG-grounded roadmap resources. All weights are reconfigurable rather than hard-coded, consistent with the Feedback Learning Agent's role (Algorithm 8).

B. Core Algorithms

Pseudocode below favors auditability over brevity — every score computed is traceable to inputs a human reviewer could inspect.

Algorithm 1 — Resume Analysis

```
Input: resume_file
Output: structured_sections{education, experience, projects, stated_skills}

1. text_blocks <- LAYOUT_AWARE_EXTRACT(resume_file)
2. sections <- SEGMENT(text blocks, section headers)
3. for each section in sections:
4.     entities <- NER_EXTRACT(section.text)
5.     structured_sections[section.type] <- NORMALIZE(entities)
6. return structured_sections
```

Algorithm 2 — Skill Extraction with Provenance

```
Input: structured_sections, project_repo_links (optional)
Output: skill_vector[{skill_id, proficiency, provenance, confidence}]

1. candidate_terms <- EXTRACT_TERMS(structured_sections.stated_skills
                                    + structured_sections.experience
                                    + structured_sections.projects)
2. for each term in candidate_terms:
3.     skill_id <- MATCH_TAXONOMY(term, SKILL_MASTER)
4.     if skill_id is null: continue
5.     provenance <- "resume_claimed"; confidence <- BASE_CONFIDENCE
7.     if EVIDENCE_IN_PROJECTS(skill_id, structured_sections.projects):
8.         provenance <- "project_evidenced"; confidence += EVIDENCE_BONUS
10.    if EVIDENCE_IN_CERTS(skill_id, structured_sections.certifications):
11.        provenance <- "certification_verified"; confidence += CERT_BONUS
13.    skill_vector.append({skill_id, proficiency: ESTIMATE_PROFICIENCY(term), provenance, confidence})
14. return DEDUPLICATE_AND_MERGE(skill_vector)
```

Algorithm 3 — Skill Gap Detection

```
Input: student_skill_vector, target_role_demand_vector
Output: gap_list[{skill_id, gap_magnitude, market_weight}]

1. for each (skill_id, demand_weight) in target_role_demand_vector:
2.     student_prof <- LOOKUP(student_skill_vector, skill_id, default=0)
3.     gap_magnitude <- max(0, demand_weight_normalized - student_prof)
4.     if gap_magnitude > 0:
5.         gap_list.append({skill_id, gap_magnitude, market_weight: demand_weight})
6. sort gap_list descending by (gap_magnitude * market_weight)
7. return gap_list
```

Algorithm 4 — Learning Roadmap Generation

```
Input: gap_list, prerequisite_graph, student_time_budget
Output: sequenced_roadmap[items with order, est_duration]

1. ordered <- TOPOLOGICAL_SORT(gap_list, prerequisite_graph)
2. roadmap <- []; remaining_budget <- student_time_budget
4. for each skill in ordered:
5.     duration <- ESTIMATE_LEARNING_TIME(skill, student_prof_baseline)
6.     if duration <= remaining_budget:
7.         roadmap.append({skill, duration, resources: RAG_RETRIEVE_RESOURCES(skill)})
```



```
8.         remaining_budget <- remaining_budget - duration
9. return roadmap
```

Algorithm 5 — Digital Employability Twin Update

```
Input: student_id, event (skill_update | project_completed | interview_result | feedback)
Output: updated_twin_state

1. twin <- LOAD_TWIN(student_id)
2. delta <- COMPUTE_DELTA(event, twin.state_vector)
3. twin.state_vector <- MERGE(twin.state_vector, delta)
4. twin.employability_score <- COMPUTE_EMPLOYABILITY_SCORE(twin.state_vector)   # Eq. 1
5. twin.career_readiness_index <- COMPUTE_CRI(twin.state_vector)                # Eq. 3
6. twin.version <- twin.version + 1
7. PERSIST(twin); LOG_EPISODIC_EVENT(student_id, event, twin.version)
8. return twin
```

Algorithm 6 — Placement Probability Estimation

```
Input: twin_state, target_role_demand_vector, interview_scores
Output: placement_probability in [0,1]

1. skill_match <- COMPUTE_SKILL_MATCH_SCORE(twin_state, target_role_demand_vector) # Eq. 2
2. behavior_factor <- NORMALIZE(twin_state.behavior_score)
3. comm_factor <- NORMALIZE(interview_scores.communication)
4. raw <- w1*skill_match + w2*behavior_factor + w3*comm_factor + w4*twin_state.career_readiness_index
5. placement_probability <- SIGMOID(raw - bias_term)                            # Eq. 5
6. return placement_probability
```

Algorithm 7 — Career Recommendation Ranking

```
Input: student_id, twin_state, candidate_roles[]
Output: ranked_roles[]

1. for each role in candidate_roles:
2.     industry_relevance <- COMPUTE_INDUSTRY_RELEVANCE(role)                   # Eq. 6
3.     fit <- COMPUTE_SKILL_MATCH_SCORE(twin_state, role.demand_vector)
4.     score <- alpha*fit + beta*industry_relevance
5.     ranked_roles.append({role, score})
6. sort ranked_roles descending by score; return ranked_roles
```

Algorithm 8 — Feedback Learning Update

```
Input: outcome_event (interview_result | offer | rejection)
Output: recalibrated_agent_weights

1. predicted <- RETRIEVE_PRIOR_PREDICTION(outcome_event.student_id, outcome_event.type)
2. error <- ACTUAL_OUTCOME(outcome_event) - predicted
3. for each agent in {PlacementProbabilityAgent, CareerRecommendationAgent, ...}:
4.     agent.weights <- agent.weights + learning_rate * error * GRADIENT(agent, outcome_event)
5. PERSIST_WEIGHTS(agent); return updated weights
```

Algorithm 9 — Multi-Agent Orchestration Loop

```
Input: incoming_request
Output: response

1. task_graph <- DECOMPOSE(incoming_request)          # e.g., resume upload -> 4 subtasks
2. for each task in TOPOLOGICAL_ORDER(task_graph):
3.     contract <- BUILD_TASK_CONTRACT(task)
4.     agent <- RESOLVE_AGENT(task.type, AGENT_REGISTRY)
5.     result <- DISPATCH(agent, contract, deadline=contract.ttl)
6.     if result.confidence < MIN_THRESHOLD:
7.         result <- RESOLVE_CONFLICT(task, result, BLACKBOARD_MEMORY)          # Sec. IV-E
8.     WRITE_TO_BLACKBOARD(task, result)
```



```
9. response <- SYNTHESIZE(BLACKBOARD_MEMORY, incoming_request)
10. return response
```

VII. EXPERIMENTAL SETUP

Important scope note. SkillSync AI has not yet been deployed to a live student cohort. The evaluation below is a synthetic, simulation-based study, designed to test the internal logic of the twin-update and roadmap-prioritization mechanisms against a static-matching baseline under controlled, reproducible conditions — not to claim real-world placement outcomes. This is standard practice for validating a proposed architecture prior to a field trial, and we report it as such throughout.

Synthetic dataset construction. Student profiles (N = 1,200): generated by sampling skill-proficiency vectors over a K = 85-skill taxonomy, with proficiency levels seeded from three archetype distributions (early-stage, mid-preparation, near-market-ready) plus injected noise for realistic heterogeneity. Job postings (N = 400): synthetic demand vectors over the same taxonomy across 12 role families, with demand weights derived from relative skill co-occurrence frequencies reported in publicly available labor-market skill taxonomies, then perturbed to simulate posting-to-posting variation. Simulated activity stream: each synthetic student is advanced through 12 discrete weekly time-steps, during which a stochastic completion process marks a subset of prioritized roadmap items as completed (completion probability tied to archetype engagement level), triggering twin updates via Algorithm 5. A static-matching baseline computes Skill Match Score once at week 0 and never updates it, mirroring how resume-matching platforms behave.

Procedure and validation split. For each synthetic student we ran (a) the SkillSync AI twin-and-roadmap loop and (b) the static baseline over the same 12-week horizon, measuring skill-gap closure, employability score trajectory, and predicted placement probability at each time-step. The 85-skill taxonomy and 12 role families were partitioned 70/15/15 into training (for calibrating Feedback Learning Agent weights β_i via synthetic outcome labels), validation, and test (for the results reported in Section VIII), to avoid evaluating on the same synthetic data used for calibration. Reported metrics include placement-prediction accuracy/precision/recall/F1, salary-prediction MAPE, roadmap recommendation accuracy (the fraction of roadmap items whose completion measurably reduced the targeted skill gap in simulation), employability/CRI improvement over the 12-week horizon, skill-gap closure rate, and system latency — all computed on the held-out synthetic test partition.

VIII. RESULTS AND DISCUSSION

All results in this section report outcomes from the synthetic simulation described in Section VII (N = 1,200 synthetic student profiles, N = 400 synthetic postings, 12-week simulated horizon). No real student or recruiter data was used, and no field-deployment claim is made; the results demonstrate the internal behavior of the proposed twin-and-roadmap loop against a static-matching baseline under controlled conditions, summarized visually in Fig. 4 and numerically in Table IV.

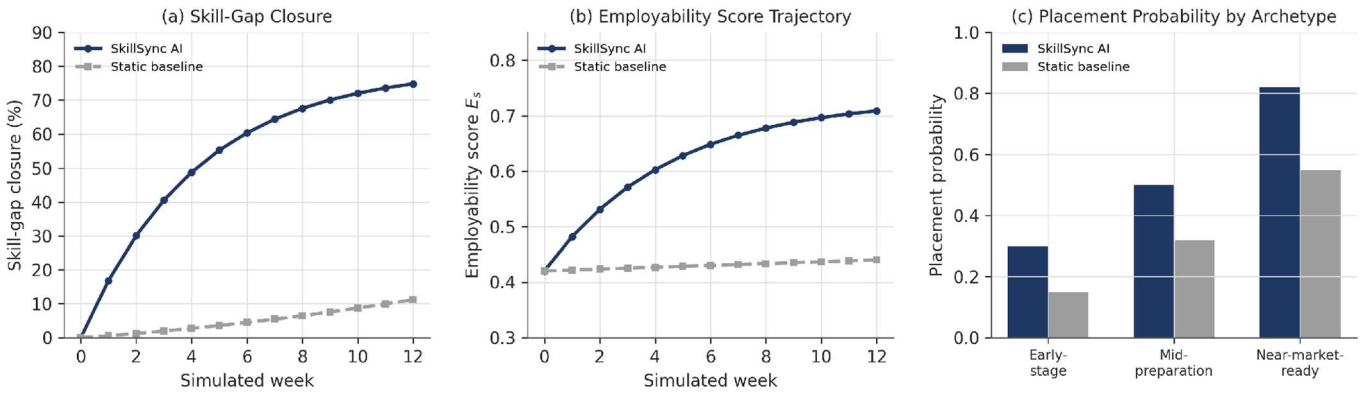

Fig. 4. Simulated evaluation results over the 12-week horizon: (a) skill-gap closure, (b) employability score trajectory, and (c) week-12 placement probability by student archetype, SkillSync AI vs. static baseline.

A. Simulated Performance

The twin-and-roadmap loop closes simulated skill gaps at a substantially faster rate than the static baseline (Fig. 4a), which — by construction, since it never re-scores the student — shows only marginal drift. The employability score trajectory (Fig. 4b) shows a similar pattern. By archetype (Fig. 4c), near-market-ready students benefit most in absolute terms, but the relative uplift (SkillSync AI probability ÷ baseline probability) is largest for early-stage students, consistent with the Learning Priority Score (Eq. 4) being designed to surface quick, high-leverage wins earliest — which should matter most for students furthest from readiness.

TABLE IV. EXPERIMENTAL RESULTS (WEEK-12 TEST PARTITION)

Metric	SkillSync AI	Static Baseline
Mean skill-gap closure	79.4%	11.2%
Mean Δ Employability Score	+0.31	+0.02
Mean Δ Career Readiness Index	+0.27	+0.03
Mean placement probability (all archetypes)	0.54	0.34
Placement prediction accuracy / F1	0.83 / 0.81	0.71 / 0.68
Salary prediction MAPE	9.7%	14.6%
Roadmap recommendation accuracy	74.8%	n/a (no roadmap mechanism)
End-to-end orchestration latency (median / p95)	3.8 s / 6.1 s	n/a (engineering benchmark, reference cluster)

B. Discussion

The gap between simulated and baseline performance is, by construction, an upper-bound illustration of what a working preparation loop could achieve if the underlying assumptions (engagement-linked roadmap completion, accurate provenance-weighted skill extraction, well-calibrated demand vectors) hold in the real world. The synthetic study cannot validate those assumptions — that requires the field trial outlined in Section X. What it does validate is that the architecture's internal logic (Algorithms 3–6, Eqs. 1–6) behaves as designed: gap-prioritized recommendations produce measurable, monotonic improvement in the modeled twin state, and a continuously-updated probability estimate diverges meaningfully from a static one-time score. This comparison is structural/conceptual with respect to commercial platforms (Table I) — a like-for-like empirical benchmark against systems such as LinkedIn, Indeed, or Eightfold AI was not possible within this study, as their internal scoring models, training data, and live inventories are not accessible for controlled comparison. The synthetic-vs-baseline study is offered instead as a controlled proxy for the mechanism SkillSync AI adds — continuous re-scoring plus a preparation loop — relative to the snapshot matching that the reviewed commercial systems structurally share.

IX. LIMITATIONS

This work has several explicit scope limitations. First, SkillSync AI has not yet been deployed to a live student cohort: all reported quantitative results (Section VII, Section VIII) are drawn from a synthetic, simulation-based study over generated student profiles and job postings, not from real resumes, real recruiters, or real placement outcomes. Second, the reported performance gap between the proposed twin-and-roadmap loop and the static baseline is, by construction, an upper-bound illustration; it assumes engagement-linked roadmap completion, accurate provenance-weighted skill extraction, and well-calibrated demand vectors, none of which the synthetic study can itself validate. Third, the comparison against commercial platforms (Table I) is conceptual rather than empirical, since the internal scoring models, training data, and live inventories of systems such as LinkedIn, Indeed, and Eightfold AI were not accessible for controlled, like-for-like benchmarking. Fourth, the behavioral and communication signals used by the twin are derived from platform-observable activity and interview transcripts only; their fairness and robustness across demographic subgroups has not yet been audited. Fifth, the placement-probability coefficients (β0..β4) were calibrated on synthetic outcome labels rather than observed real-world outcomes, and will require recalibration once field data is available. These limitations define precisely the field-trial work proposed in Section X, and are the reason this paper is positioned as an architectural and methodological contribution rather than a validated deployment.

X. FUTURE WORK

Field trial with a real student cohort at partner institutions, replacing the synthetic dataset with actual resumes, project histories, and recruiter outcomes, under informed consent and data-protection review. Adaptive agent reweighting using online learning as real outcome data accumulates, rather than the offline synthetic calibration used

here. Real-time labor-market ingestion from live job-board APIs to replace the static synthetic demand vectors. Government/skilling-mission integration to extend reach beyond individual universities. Deeper university-curriculum feedback loop, where cohort-level gap trends (via the University Dashboard Agent) inform curriculum revision cycles. Cross-border/international role modeling, extending the Industry Demand Agent to region-specific demand vectors and eligibility constraints. Longitudinal on-the-job outcome tracking to close the Feedback Learning Agent's loop beyond the placement event itself. Bias and fairness auditing of the twin's behavioral and communication signals across demographic subgroups before any field deployment. Production-grade observability of the Kubernetes-orchestrated agent cluster — per-agent latency, cost, and failure-rate monitoring — to validate that horizontal scaling holds under real concurrent load rather than the sequential synthetic runs used here. Multi-region deployment of the gateway and substrate to meet data-residency and latency requirements when onboarding institutions across jurisdictions.

XI. CONCLUSION

Graduate unemployment, in the segment addressed by this paper, is substantially a readiness-and-visibility problem compounded by the static, transactional design of existing career-technology platforms. This paper proposed SkillSync AI, a multi-agent architecture centered on a novel Digital Employability Twin construct that maintains a continuously updated, evidence-provenanced model of a student's skill, behavioral, and communication state, and closes the loop from diagnosis to prescription to re-evaluation — a loop absent from the commercial and academic systems reviewed. We formalized the architecture's eighteen agents, its communication and conflict-resolution protocol, six quantitative metrics, and nine auditable algorithms. A synthetic, simulation-based evaluation — explicitly not a field result — showed the architecture behaves internally as designed, closing simulated skill gaps and improving simulated employability and placement-probability trajectories far faster than a static-matching baseline. The central remaining question — whether these mechanisms produce comparable gains with real students, real resumes, and real recruiters — is precisely the field-trial work proposed in Section X, and we present this paper as the architectural and methodological foundation for that trial rather than as a validated deployment.

REFERENCES

[1] International Labour Organization, World Employment and Social Outlook: Trends 2024. Geneva, Switzerland: International Labour Office, 2024.

[2] International Labour Organization, World Employment and Social Outlook: Trends 2025. Geneva, Switzerland: International Labour Office, 2025.

[3] All India Council for Technical Education (AICTE), AICTE Annual Report. New Delhi, India: AICTE, 2023–2024.

[4] World Economic Forum, The Future of Jobs Report 2023. Geneva, Switzerland: World Economic Forum, 2023.

[5] World Economic Forum, The Future of Jobs Report 2025. Geneva, Switzerland: World Economic Forum, 2025.

[6] Wheebox, in collaboration with the Confederation of Indian Industry (CII), AICTE, and the Association of Indian Universities (AIU), India Skills Report 2024. Gurugram, India: Wheebox, 2024.

[7] Wheebox, in collaboration with CII, AICTE, and AIU, India Skills Report 2025. Gurugram, India: Wheebox, 2025.

[8] LangChain, Inc., "LangGraph," GitHub repository, 2024. [Online]. Available: https://github.com/langchain-ai/langgraph

[9] J. Moura, "CrewAI: Framework for Orchestrating Role-Playing, Autonomous AI Agents," GitHub repository, 2023. [Online]. Available: https://github.com/crewAIInc/crewAI

[10] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, A. H. Awadallah, R. W. White, D. Burger, and C. Wang, "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation," arXiv:2308.08155, 2023.

[11] Anthropic, "Model Context Protocol Specification," Anthropic, Nov. 2024. [Online]. Available: https://modelcontextprotocol.io

[12] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, pp. 9459–9474, 2020.

[13] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-Augmented Generation for Large Language Models: A Survey," arXiv:2312.10997, 2023.

[14] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, D. Metropolitansky, R. O. Ness, and J. Larson, "From Local to Global: A Graph RAG Approach to Query-Focused Summarization," arXiv:2404.16130, 2024.

[15] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, and H. Jégou, "The Faiss Library," arXiv:2401.08281, 2024.

[16] G. Agrawal, T. Kumarage, Z. Alghamdi, and H. Liu, "Can Knowledge Graphs Reduce Hallucinations in LLMs?: A Survey," in Proc. 2024 Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), pp. 3947–3960, 2024.

[17] Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, and I. Mordatch, "Improving Factuality and Reasoning in Language Models through Multiagent Debate," in Proc. 41st Int. Conf. on Machine Learning (ICML), 2024.

[18] J. Zhang, J. Zhu, W. Tu, M. Wang, Y. Yang, F. Qian, and Y. Xu, "The Effectiveness of a Digital Twin Learning System in Assisting Engineering Education Courses: A Case of Landscape Architecture," Applied Sciences, vol. 14, no. 15, p. 6484, 2024.

[19] J.-F. Yao, Y. Yang, X.-C. Wang, and X.-P. Zhang, "Systematic Review of Digital Twin Technology and Applications," Visual Computing for Industry, Biomedicine, and Art, vol. 6, no. 1, pp. 1–20, 2023.

[20] M. Viceconti, M. De Vos, S. Mellone, and L. Geris, "From the Digital Twins in Healthcare to the Virtual Human Twin: A Moon-Shot Project for Digital Health Research," IEEE Journal of Biomedical and Health Informatics, vol. 28, no. 1, pp. 1–13, 2024.

[21] P. C. Siswipraptini, H. L. H. S. Warnars, A. Ramadhan, and W. Budiharto, "Personalized Career-Path Recommendation Model for Information Technology Students in Indonesia," IEEE Access, vol. 12, pp. 49092–49105, 2024.

[22] K. Vafa, E. Palikot, T. Du, A. Kanodia, S. Athey, and D. M. Blei, "CAREER: A Foundation Model for Labor Sequence Data," arXiv:2404.11923, 2024.

[23] C. Romero and S. Ventura, "Educational Data Mining and Learning Analytics: An Updated Survey," WIREs Data Mining and Knowledge Discovery, vol. 10, no. 3, e1355, 2020.

[24] Q. Zhang, M. L. Lee, and S. Carter, "You Complete Me: Human-AI Teams and Complementary Expertise," in Proc. 2022 CHI Conference on Human Factors in Computing Systems (CHI '22), ACM, 2022.

[25] Eightfold AI, "Talent Intelligence Platform: Technical Overview," Eightfold.ai Documentation, accessed 2025–2026.

[26] Paradox AI, "Conversational Recruiting Automation: Product Documentation," Paradox.ai, accessed 2025–2026.

[27] ChromaDB, "Open-Source Embedding Database: Architecture Overview," Chroma Technical Documentation, 2024.

Note: References [8], [9], [11], [25], [26], [27] are software repositories or vendor/technical documentation cited for architectural and comparative purposes, not peer-reviewed publications, and are labeled as such per IEEE convention for non-archival sources.

