



A Mastery-Gated AI Framework Bridging Diagnostic Learning Paths to Verified Career Deployment

Sivasangkari K

Department of Artificial Intelligence and machine learning
SNS College Of Technology
Coimbatore
sivasangkari1111@gmail.com

Roobini N

Department of Artificial Intelligence and machine learning
SNS College Of Technology Coimbatore
roobini.natarajan005@gamil.com

Pallavi C

Department of Artificial Intelligence and machine learning
SNS College Of Technology Coimbatore
pallavichandrapa07@gmail.com

Rithika S

Department of Artificial Intelligence and machine learning
SNS College Of Technology
Coimbatore
rithikasathishh987@gmail.com



<https://doi.org/10.55041/ijssmt.v2i10.079>

Cite this Article: K, S., N, R., C, P. & S, R. (2026). A Mastery-Gated AI Framework Bridging Diagnostic Learning Paths to Verified Career Deployment. *International Journal of Science, Strategic Management and Technology*, 02(10), 1-9. <https://doi.org/10.55041/ijssmt.v2i10.079>

License:  This article is published under the Creative Commons Attribution 4.0 International License (CC BY 4.0), permitting use, distribution, and reproduction in any medium, provided the original author(s) and source are properly credited.

Abstract: This engineering case study examines CareerAI as a mastery-gated framework linking role-based learning, diagnostic skill feedback, staged assessment, interview rehearsal, digital certificates, and outbound job discovery. The supplied implementation inventory describes eight technical career pathways with eight topics per role, three gated Learn-Try-Build phases, three five-item assessment stages, adaptive prioritization of historically weak skills, a six-part interview simulator including voice interaction and sandboxed JavaScript execution, and owner administration. React, Express, SQLite, and browser storage divide the system into presentation, protected persistence, and client-side progress concerns. Question order is randomized with Fisher-Yates shuffling; scores and readiness levels use disclosed deterministic thresholds; interview answers are matched with concept dictionaries. No trained machine-learning system, live job-feed API, or cryptographically signed credential is established by the evidence. Thus, 'verified career deployment' is a proposed objective, not an observed placement or independently validated skill outcome. The provided inventory reports 19 passing automated tests, but an executable repository, raw execution logs, and participant studies were unavailable for independent verification; nine retained user-supplied screenshots were available as illustrative interface-state evidence. We therefore analyze documented functionality, mathematical boundary behavior, data and trust boundaries, and a reproducible testing protocol without claiming unmeasured learning gains or predictive accuracy. The contribution is a traceable system design, a formalization of its decision rules, and a critical identification of validity, usability, cross-device consistency, and security requirements for subsequent empirical work.

Index Terms: career learning, formative assessment, rule-based adaptation, interview simulation, learning analytics, digital credentials, reproducibility, web applications.

I. INTRODUCTION

A. Motivation and Educational Setting

Technology-career preparation is not one decision. A learner identifies a possible occupation, translates that occupation into a sequence of knowledge and practice goals, completes exercises, checks current competence, rehearses communicating knowledge, and finally looks for an entry point into employment. Each activity may be supported by a different website; however, the learner is responsible for connecting evidence across those sites. A watched tutorial does not automatically explain which interview skill it addresses, and a quiz score may not show what should be studied next. CareerAI addresses this gap through role-specific learning paths,

diagnostic-style weakness feedback, and application-defined progression gates before interview practice and job discovery.

Feedback theory provides a reason to keep evaluation near learning. Hattie and Timperley discuss how feedback can influence learning depending on the information conveyed [1], and Nicol and Macfarlane-Dick relate feedback practice to the learner's capacity to regulate future work [2]. CareerAI uses these ideas as design motivation rather than as proof that its own feedback is pedagogically effective. The documented workflow makes progress and weak areas visible to the learner, but there is no reported educational intervention demonstrating that these features improve independent test performance.

The application is aimed at novice and intermediate learners preparing for technology roles. It is not represented as an accredited training provider, job-placement service, or licensed psychological assessor. A digital completion passport is an application credential and cannot be equated with an institutionally certified professional qualification. Accordingly, neither the 'Job Ready' label nor the title's 'verified career deployment' language constitutes independent proof of competence or employment placement.

B. Problem Statement

Let a career learner have a selected role r , a set of topics $T(r)$, a sequence of phase-completion states L , an assessment history A , and interview observations I . The engineering problem is to maintain an understandable, mastery-gated progression from r to $T(r)$ through the later outcomes without losing the relationship between the actions and the evidence used to authorize each step. A secondary problem is selecting useful practice based on prior mistakes while avoiding an exaggerated claim that simple sorting is equivalent to psychometrically adaptive testing or machine learning.

In conventional site-by-site preparation, there is no guaranteed shared user identity, no common record of what has been attempted, and no agreed meaning of completion. CareerAI addresses this through a learner dashboard and a local state machine for progress gates. The resulting system can make the sequence explicit and demonstrable, although the evidence supplied does not establish that the resulting recommendations are optimal, nor that a course mark is valid across devices.

The observed scope is described by an implementation inventory rather than a checked-out source snapshot. The distinction between a specified mechanism, a reported test

outcome, and an independently observed browser interaction is maintained throughout the manuscript. This precaution avoids converting a detailed engineering description into an unsubstantiated performance experiment.

C. Objectives and Research Questions

The first objective is to formalize CareerAI's actual reported system design: user types, technology layers, storage boundaries, named modules, interfaces, input validation, and progression dependencies. The second is to reconstruct the deterministic algorithms sufficiently precisely that a third party can inspect arithmetic boundary cases. The third is to record which functionality is supported by supplied documentation and define the additional observations needed before claims of learning benefit, security assurance, or credential reliability could be made.

Four research questions structure the analysis. RQ1 asks which end-to-end learner and owner workflows are represented in the documented implementation and what their control boundaries are. RQ2 asks how assessment history, scoring penalties, and interview concept matching determine feedback and access. RQ3 asks which claims can be sustained by the currently supplied functional evidence and which require new execution or human evaluation. RQ4 asks what design limitations arise when browser-local progress is combined with protected server-side course marks and a client-produced completion document.

D. Contribution and Novelty Boundary

The contribution is principally the integration and analysis of a diagnostic-to-career workflow: a role-specific curriculum, gated practice, formative testing, structured interview rehearsal, descriptive skill feedback, credential generation, and career-specific outbound search are connected within one application. It is also a reproducibility contribution at the document level: important state transitions, score formulas, test assertions, data relationships, and weaknesses are described in verifiable terms rather than marketing phrases.

No claim of inventing adaptive testing, credential hashing, formative assessment, or browser code execution is made. The application is not shown to outperform any prior system; there are no common datasets or head-to-head experiments. Its value for subsequent research lies in being a concrete, traceable platform on which correctness, learner experience, and measurement quality can be studied separately.

II. RELATED WORK AND RESEARCH GAP

A. Feedback, Self-Regulation, and Mastery

Zimmerman's account of self-regulated learning describes learner agency, goal setting, monitoring, and reflection as connected behaviors [3]. CareerAI's staged activity design reflects some of those concerns: each topic is subdivided into an exposure phase, a practice phase requiring a written reflection, and an artifact-oriented build phase. However, a completion checkbox does not demonstrate conceptual change. The implementation records a user's declared or interaction-triggered activity completion and awards fixed points, not an independently observed mastery score.

Shute's review of formative feedback distinguishes information intended to improve thinking from a simple grade display [4]. This distinction is relevant to the application's strongest- and weakest-skill summaries. A claim such as 'weakest skill: JavaScript' becomes actionable only when the category is supported by enough representative items and linked to an appropriate next task. The current reported implementation provides such links through curated suggestions, but available evidence does not establish their appropriateness for diverse learner backgrounds, nor whether learners follow them.

These foundations therefore support the selection of design concerns—transparent goals, feedback specificity, and reflection—but do not serve as borrowed proof of CareerAI's effectiveness. The unit of analysis in this manuscript is the documented software workflow, not the causal impact of a learning intervention.

B. Adaptive Testing Versus Weakness-Priority Selection

Meijer and Nering describe computerized adaptive testing in terms of item selection, latent-trait estimation, exposure, precision, and item-bank construction [5]. CareerAI uses a much narrower adaptation mechanism: it estimates error ratios for skill tags from stored prior answers, prioritizes tags with the poorest ratio in later stages, and randomly shuffles questions and options. It does not report item-response calibration, difficulty estimation, measurement uncertainty, or stopping rules based on score precision. Calling it a 'weakness-prioritized question selector' is therefore more technically accurate than calling it a fully calibrated computerized adaptive test.

Dai, Gu, and Zhu studied recommendations derived from adaptive testing technology and evaluated learner performance in a classroom comparison [6]. Their design ties item difficulty and learner ability to a common scale; the documented CareerAI algorithm has neither parameter. The contrast clarifies what would be needed for a stronger claim: rated items, a large response dataset, reliable ability estimation, and a carefully controlled evaluation against a nonadaptive baseline.

Randomized option order is also not equivalent to a measured anti-cheating outcome. Fisher–Yates shuffling reduces predictability of answer positions, while tab-away monitoring detects some browser focus changes. Neither establishes whether collusion, external devices, or screenshots are prevented. The manuscript describes these as interface-level deterrents rather than verified examination integrity.

C. Context-Aware Recommendations and Career Guidance

Abu-Rasheed, Weber, and Fathi survey contextual indicators for personalized learning recommendations and emphasize that pedagogical context includes more than a single test score [7]. CareerAI uses a limited set of signals: selected role, past skill-level responses, phase progress, and readiness categories. It does not document time availability, formal prior qualifications, labor-market skill demand, protected demographic dimensions, or longitudinal learning patterns as model inputs. This simplification makes the rules interpretable but limits personalization.

Kamal and co-authors synthesize systems supporting academic choices in higher education and report a range of recommender strategies and evaluation settings [8]. CareerAI differs in targeting a continuous preparation workflow after a role has been chosen, not merely advising which course or institution to select. Unlike recommenders evaluated with ranking metrics, its job component creates encoded search queries for external services. It cannot reasonably be assigned precision at k or recommendation accuracy without an itemized relevance dataset and a retrieval/ranking algorithm.

The recent career-guidance scoping review by Manganello and colleagues observes the broad interest in data-informed choices [9]. CareerAI is positioned as a transparent rules-based implementation within this domain, not as evidence that machine-learning career counseling is superior or that its suggestions predict employment outcomes. The lack of a participant dataset is a central gap between the motivation for such tools and substantiated claims about their effectiveness.

D. Learning Analytics, Digital Identity, and Security

Oliva-Cordova, Garcia-Cabot, and Amado-Salvatierra review the use of learning analytics to support teaching practice [10]. CareerAI's analytics are learner-facing and predominantly descriptive: percent correct per skill, an overall average across passed stages, and categorical readiness labels. Without intervention logs or item-level response distributions, the displays cannot establish causal learning progress. Labels should be interpreted as internal dashboard summaries with disclosed scale construction.

The W3C Verifiable Credentials Data Model v2.0 formalizes issuer–holder–verifier relationships and integrity requirements for portable claims [11]. CareerAI's application passport reportedly includes a repeatable identifier derived from FNV-1a. Such an identifier assists lookup or consistent naming but is neither a cryptographic digital signature nor evidence that an external verifier can check issuance status. This is an important terminology distinction: 'downloadable certificate' is supported; independently verifiable credential issuance is not yet established.

The OWASP Application Security Verification Standard provides checkable expectations for authentication, session management, and protected actions [12]. CareerAI reports password hashing and protected user/admin cookies, but only selected controls are documented. In the absence of source-based review and penetration testing, the manuscript does not make a broad security certification claim. UNESCO's guidance on human-centered educational AI also supports caution when educational interfaces present strong automated judgements about individuals [13], even though the current implementation is deterministic.

E. Cross-Study Comparison and Explicit Research Gap

TABLE I. COMPARATIVE POSITIONING OF SOURCE-SUPPORTED STUDIES AND CAREERAI

Prior work	Primary focus / evidence	CareerAI contrast
Hattie–Timperley [1]; Shute [4]	Feedback research and conceptual synthesis	Shows feedback pages; effect not measured

Prior work	Primary focus / evidence	CareerAI contrast
Zimmerman [3]; Nicol–Macfarlane–Dick [2]	Self-regulated learning and feedback principles	Staged goals and reflection; no intervention study
Meijer–Nering [5]	Psychometric adaptive testing foundations	Uses error-ranked question tags, not IRT
Dai et al. [6]	Difficulty-aware recommendation with classroom comparison	No calibrated difficulty or causal comparison
Abu-Rasheed et al. [7]	Context indicators for adaptive learning	Uses narrow role and error-history context
Kamal et al. [8]	Review of academic-choice recommender systems	Offers career workflow but no ranked live job feed
W3C [11]; OWASP [12]	Credential assurance and application controls	Local hash ID and partial controls; further assurance needed

The research gap addressed here is practical rather than a demonstrated algorithmic breakthrough: published work often isolates feedback, item selection, advising, or trust, whereas CareerAI's documented workflows put multiple such activities into an explicit progression contract for career practice. The engineering question is whether that contract is internally consistent, observable, and defensible. The much larger unanswered empirical question is whether connected guidance changes learner performance or employment readiness relative to available alternatives.

Table I separates theoretical and methodological literature from application behavior. A blank or unreported feature in a cited paper should not be interpreted as proof of absence. Methods, populations, outcome metrics, and evaluation conditions differ too much to rank these studies and the project by a common performance number. The work makes no claim of being the first career-preparation website or of delivering measured superiority.

III. PROBLEM FORMULATION AND SYSTEM REQUIREMENTS

A. Users, Inputs, Outputs, and State

Two authenticated user roles are described. A learner registers or signs in, chooses a role, navigates curated topic resources, completes practice reflections and assessments, interviews, reviews analytics, and downloads a certificate when eligibility conditions are satisfied. An owner administrator signs in through separate routes and reviews registrations, activity history, and completed course marks. These accounts use different authentication contexts and session-cookie names to reduce accidental role confusion.

The input surface includes credentials, selected career identifiers, exercise responses, multiple-choice selections, visibility events, interview answers, optional speech transcripts, coding strings, location queries, and profile updates. Derived outputs include topic status, experience points, streak counts, scores, readiness descriptors, interview feedback, downloadable PDF content, audit events, and outbound search URLs. The implementation inventory is specific about several derived fields but not about the exact schema of every client-side progress object. Consequently, the document reports the stated structure and does not invent unnamed database columns.

A crucial conceptual distinction is between an event, a claim, and an assessed outcome. Clicking a 'completed' control is an application event; a stored topic-completion flag is the application's recorded claim; passing an independently administered skills test would be a learning outcome. CareerAI currently supplies the first two kinds of observation for several modules. Conflating them would overstate evidence.

B. Functional Requirements and Acceptance Conditions

TABLE II. DOCUMENTED REQUIREMENTS AND VALIDATION BOUNDARIES

ID	Requirement/ expected behavior	Verification channel
FR-01	Create and authenticate separate learner accounts	Server test + browser
FR-02	List 8 supported role paths with 8 topics each	Data inspection + UI
FR-03	Unlock Try only after Learn; Build only after Try	Unit tests + state transitions
FR-04	Deliver 5 items per assessment stage with shuffled options	Unit tests + browser
FR-05	Apply 60% pass and focus-warning deductions	Boundary test cases
FR-06	Gate interview until 3 assessment stages passed	Route and state validation
FR-07	Provide six-part interview and sandboxed JS challenge	Browser integration
FR-08	Release passport only after assessment + interview pass	Authorization tests
FR-09	Generate role-filtered external job search URLs	URL parsing + link test
FR-10	Restrict owner audit information to administrators	Protected API integration

Each functional requirement needs an oracle: an independently stated expected result for a known input. For example, submitting correct answers on three of five assessment items with exactly two tab changes should produce a score of 60 and unlock the next stage, whereas one additional chargeable warning should reduce the score to 55. A system that simply displays the score but leaves later routes accessible would not satisfy the progression requirement. The specified behavior should be tested both through normal navigation and by directly visiting protected URLs.

Requirements also imply retry behavior. A failed stage must remain retryable without resetting unrelated completed learning topics. The documentation states that questions and options are randomized per attempt and later stages prioritize previously weak skills. Such behavior requires careful separation between past attempt history and the new attempt's current answers. Without this separation, adaptation could incorrectly use a partial attempt as prior evidence.

C. Nonfunctional, Trust, and Usability Requirements

Performance concerns include initial page load, API latency, assessment transition speed, code-execution responsiveness, and the overhead of synchronous SQLite access. Since no independent timing traces were supplied, the manuscript defines relevant measures without assigning them numerical values. Reliability includes persistence under page reload, browser back/forward navigation, interrupted sessions, and

concurrent login. Cross-device continuity is especially constrained because many active progress values are reportedly stored in localStorage rather than in the backend.

Security concerns center on identities, roles, stored password hashes, authentication cookies, course-result modification, arbitrary code execution, and personal activity logs. The existence of a security primitive is not evidence that the system passes a comprehensive assessment. For example, a signed cookie may protect against tampering but not necessarily guarantee immediate revocation without server state; a time limit on a code runner does not by itself prove that every nonterminating script can be stopped; a hashed identifier is not a certificate signature.

Accessibility and inclusivity matter for keyboard interaction, contrast, screen-reader announcements, understandable voice controls, and low-bandwidth use. The speech interface depends on browser support and language settings. A design that demands voice input without a text alternative would exclude some users; the project reportedly provides typed answers as an alternative, but no WCAG conformance report, assistive-technology audit, or user study is available.

D. Trust Boundaries and Invariants

The application crosses three distinct trust boundaries. The browser is the interactive surface and stores progress records that the user can potentially manipulate. The Express API validates identities and selected server writes. SQLite provides durable account, audit, and course-result storage on the server. Outbound job platforms constitute a fourth-party boundary: visiting them transfers the learner to an external service with separate privacy and availability policies.

A suitable invariant for protected course marks is that the learner identity comes from the validated session rather than a user ID supplied in a request body. Another is that an administrator may read organization-wide audit data only after owner authorization. For certificate eligibility, the stronger invariant would be that the server—not only modifiable client-side state—authoritatively confirms all prerequisite passes. The inventory confirms protection for the course-results endpoint but does not establish this stronger end-to-end credential invariant.

Requirements should remain testable rather than aspirational. 'Secure' and 'AI powered' are too broad without a threat model and a verifiable method. This case study therefore maps each important promise to one or more concrete behaviors, records the type of evidence supporting them, and reserves unsupported promises for future work.

IV. SYSTEM ARCHITECTURE AND DATA DESIGN

A. Deployment and Runtime Topology

The supplied package inventory identifies CareerAI version 1.0.0 as a React single-page application served by Vite on local development port 4173 and an Express API running on localhost port 3001. HTTP calls beginning with /api are proxied by Vite to Express; browser requests attach same-origin credentials so that HttpOnly cookies participate in authenticated calls. Production deployment topology, TLS termination,

reverse-proxy hardening, backup process, and external hosting have not been specified, so the system is evaluated as a local web application rather than an internet-scale production service.

Node.js 22 or newer is the documented runtime. React 18 components render learning and admin pages, React Router 7 controls client navigation and route guards, and application state is shared through AuthContext, AdminAuthContext, and CareerContext. The API centralizes authentication and course-record requests. The SQLite database uses better-sqlite3 with foreign-key checks and write-ahead logging enabled according to the inventory. These decisions minimize operational dependencies but also mean the single-node server and local database are natural bottlenecks under concurrent load; no throughput benchmark is reported.

Fig. 1 separates interactive, trusted, and third-party resources. The dashed conceptual trust boundary is the boundary between the browser's mutable JavaScript state and the API's authorized actions. Local drafts, XP, or unfinished answers stored in browser storage are not equivalent to a signed server grade. External job boards are search destinations and not database dependencies; their content can change independently of CareerAI.

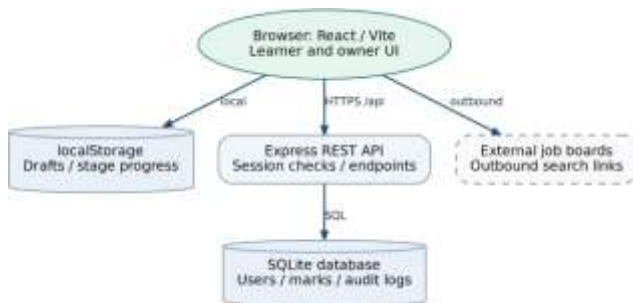


Fig. 1. Documented development architecture: React/Vite client, Express API, SQLite persistence, browser-local state, and outbound job search.

B. Frontend Organization and State Contexts

The frontend module map includes public landing, login, registration, password-help, and role-catalog pages, followed by authenticated dashboard, learning, roadmap, assessment, interview, analysis, certificate, jobs, and profile pages. React Router protects selected learner destinations; an admin-specific protected route gates owner pages. Lazy-loaded route components are reported in the inventory, but measured bundle sizes and loading behavior are not available. The layout layer supplies sidebar navigation and shared dashboard structure.

The three contexts have different responsibilities. AuthContext maintains authenticated learner identity and is involved in legacy-account migration. AdminAuthContext manages the distinct owner session. CareerContext coordinates role selection, assessment scores, warnings, and active progression. This separation is relevant because a learner should not become an owner simply because they can write a browser storage key. API-level owner guards are required even when the React interface hides the administrator destination.

Frontend components may issue optimistic visual updates while the source of truth remains either localStorage or a server

response. A progress bar is computed from locally stored phase flags; a returned server course mark is a separate record. Failure behavior must reconcile stale local views after a server write error. The inventory shows a centralized apiRequest wrapper, but it does not provide captured traces proving that all write errors are propagated consistently.

C. REST Interfaces and Resource Protection

TABLE III. PRIMARY DOCUMENTED API CONTRACTS

Group	Endpoint examples	Principal state / purpose
Health	GET /api/health	Service availability status
User auth	POST /api/auth/register, /login, /logout	Learner cookie + audit events
Session	GET /api/auth/me	Read current learner identity
Profile	PATCH /api/auth/profile; GET /api/profile	Name and course marks
Recovery	POST /api/auth/forgot-password, /reset-password	Demo password update
Course	POST /api/course-results	Upsert protected marks
Owner	POST /api/admin/login; GET /api/admin/me	Distinct owner cookie
Owner read	GET /api/admin/dashboard, /users, /users/:id	Counts and audit records
Audit	GET /api/admin/login-history	Paginated sign-in history

Table III is an inventory of the named API paths, not a newly generated OpenAPI contract. The input schemas, response codes, rate limits, and pagination caps require code inspection for formal verification. The server reportedly caps JSON bodies at 16 KB and validates name, email, and password lengths. User IDs for protected /api/profile and /api/course-results calls are derived from the signed cookie, which avoids blindly trusting a client-provided account identifier.

Authentication uses scrypt-derived passwords, a randomly generated salt, HMAC-SHA256 session signatures, and timing-safe comparisons as reported by backend/passwords.js and backend/app.js. The cookies are designated HttpOnly and SameSite=Lax, with a conditional Secure attribute in production. The documented seven-day expiration provides a lifetime bound; nevertheless, expiration, logout invalidation, renewal behavior, CSRF defenses for state-changing endpoints, and secret rotation need direct security tests under the actual deployment settings.

D. Database Schema and Durable Entities

The SQLite schema describes six tables. users stores a unique case-insensitive email, a scrypt password hash, names, and timestamps; course_results stores one completed mark per user/course combination and upserts on repetition. admin_users contains independently hashed owner credentials. login_history tracks successful, failed, and logout events for users and admins. user_activities records named events such as account creation and profile edits. password_reset_tokens has fields for token hashes, expiration, and use status, although the documented reset flow operates as a local demonstration without external email delivery.

Referential integrity constrains course results and activity records to valid user identities. For grades the documented

database check is marks between 0 and 100. These constraints help preserve internal data coherence; they do not prevent false marks if untrusted client data are accepted without a server-verified assessment attempt. The record of completion and the mathematical computation that produced the grade should be distinct auditable events in any future verified implementation.

Fig. 2 describes the main foreign-key relationships. The diagram is derived from the textual schema, not extracted from a live SQLite introspection command. Index definitions, migrations, transaction isolation under concurrency, retention rules, and encryption-at-rest status are not established by the available evidence.

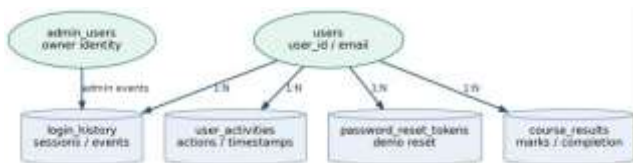


Fig. 2. Documented SQLite entity inventory and identity relationships. Reference fields are reported from project documentation, not a live schema inspection.

E. Browser-Local Storage and Migration

The inventory names a per-user storage namespace of the form `careerai_progress_<email>`, holding topic drafts, active stage outcomes, answered question IDs, anti-cheat warning counts, and interview state. This is convenient for lightweight development and offline-like continuity in one browser, but it has structural limits. Data may be lost when browser storage is cleared; different devices do not share it; and local values may be modified through developer tools. The account's final course records are reportedly persisted to SQLite, resulting in a split model of truth.

A legacy migration path is described for older plaintext accounts previously stored in `localStorage`. During login, the client detects such accounts and attempts to transfer them to the backend. The conversion should be limited to a controlled, once-only path with conflicts and partial failures handled explicitly, especially when duplicate email identities exist. The supplied inventory asserts migration behavior but contains no logged demonstration of its conflict policy, rollback, or deletion of legacy plaintext traces.

This hybrid architecture should be presented in the user interface as a meaningful data limitation. Learners can reasonably expect course credit to follow their authenticated identity across devices; the documentation supports this for stored course marks, not for all in-progress reflections, attempts, and streaks. A future server-backed progress API could reduce this discrepancy while introducing new concurrency and privacy obligations.

F. Event Sequence and Failure Propagation

The expected authenticated sequence begins with a browser POST for registration or sign-in. The Express layer validates fields and credentials and responds with a signed session cookie; later protected calls read the cookie and derive identity from it. When a passed interview triggers POST `/api/course-results`, the server validates the authenticated learner and mark bounds before updating the database. A successful response

should be observable by a subsequent GET `/api/profile` call, ideally from a fresh session as well as the active browser.

Failure conditions include network disconnection between the proxy and Express, expired cookies, database permission errors, simultaneous updates, unsupported browser speech APIs, and a blocked third-party job board. These faults occur across different ownership domains. The interface should make a write failure visible rather than silently marking a persisted award; similarly, a job-board redirect error should not invalidate a passed assessment. The documentation identifies where these boundaries exist but does not contain captured status-code traces for all cases.

V. FORMALIZED METHODS AND ALGORITHMS

A. Learning Phase Dependencies and Experience Points

For each career topic, the learning engine reportedly defines ordered phases Learn (L), Try (T), and Build (B). Completing Learn grants 25 experience points (XP) and unlocks Try. Try requires an interactive exercise and a reflection of at least 16 characters and awards another 25 XP when completed. Build requires a practical mini-project with confirmation and awards 50 XP. The total of 100 XP per completed topic is a motivational progress unit, not a standardized psychometric measure of knowledge.

Let I_L , I_T , and I_B be binary validated phase completion indicators for a topic. Equation (1) defines its awarded XP; equation (2) sums the completed topic XP across m tracked topics. The prerequisite constraints require $I_T \leq I_L$ and $I_B \leq I_T$. An XP system that increments every time a user revisits a completed phase would overcount; a correct implementation should award once per transition from incomplete to complete. The supplied tests reportedly include this kind of normalization and completion-order logic, though their fixture inputs were not supplied.

$$X_{\text{topic}} = 25I_L + 25I_T + 50I_B \quad (1)$$

$$X_{\text{total}} = \sum_{t=1}^m X_{\text{topic},t} \quad (2)$$

A topic state can be labeled Not Started, Learning, Practicing, or Mastered by examining the phase indicators. A daily streak is computed by walking backward across unique calendar dates of completed actions, starting from today or yesterday. The timezone used for interpreting calendar dates matters: a user near midnight may see a streak change when moving devices or timezone. Streak length should not be equated with hours studied, because a single recorded action is enough to mark a date in the stated algorithm.

The curriculum content is curated rather than generated in real time. The files `learning.js` and `roadmaps.js` reportedly contain tutorial URLs, readings, core concepts, exercises, and mini-project definitions. Content coverage, link survival, copyright permissibility, and topic-to-role validity all require periodic subject-matter review. An unavailable external tutorial is a resource failure, not proof that a learner has failed the phase.

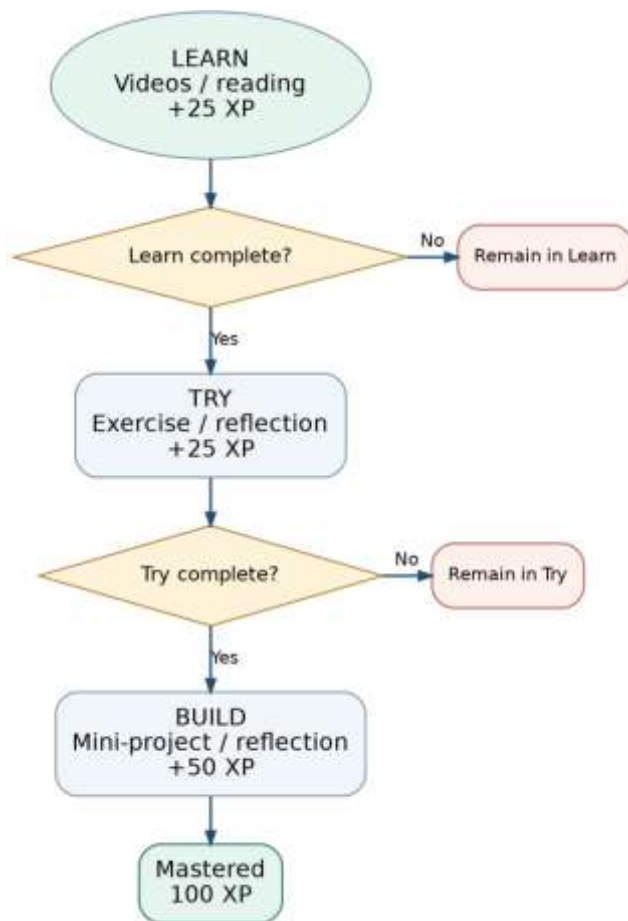


Fig. 3. The three learning phases and their documented unlock and XP allocation rules.

B. Historical Weakness Estimation

Question records are tagged with skills. For a skill k with n_k historical scored responses and c_k correct responses, the documented selector uses the correctness ratio in equation (3). A lower ratio indicates relatively poorer performance on the stored items, so Stage 2 and Stage 3 prioritize questions from the three weakest skill tags. The rule is transparent, but it is only an empirical proportion over the recorded items. It is unreliable when n_k is very small, when item difficulty differs across tags, or when repeated questions inflate the denominator.

A missing history requires a defined fallback so that $n_k = 0$ is not divided into c_k . The inventory does not specify the exact tie-break behavior. A rigorous reproduction must test equal ratios, empty histories, completely correct histories, previously seen items, and shortages of eligible questions. If the selection pool contains fewer than five qualifying items, a deliberate fallback should guarantee that the stage still has five items without duplicating conflicting answer keys.

The system description states that previously unused and used questions are sorted with weak-skill items prioritized. This is best interpreted as heuristic question selection, not a latent-ability estimator. A skills tag is a content label, while an adaptive-testing skill estimate would need item calibration and a specified measurement model.

$$a_k = \frac{c_k}{n_k}, \quad n_k > 0 \quad (3)$$

C. Randomization and Question Presentation

The question selector reportedly applies Fisher–Yates shuffling to candidate questions and each question's answer options. For an array of length n , iteration moves backward from $i = n-1$ to 1 and exchanges element i with an element indexed by j , where j follows equation (4). When U is uniform on $[0,1)$, every valid index between zero and i is reachable. If the implementation uses `Math.random`, it is appropriate for interface randomization but does not provide cryptographically secure unpredictability.

The correct answer must remain logically attached to its option text after shuffling. A common error is to preserve the pre-shuffle correct index while changing displayed option positions. Reported assessment tests check answer-key validity, but the project files were not independently inspected. Additional tests should compare the answer text before and after every randomized permutation, verify stable scoring for all positions, and confirm that the shuffled array has no lost or repeated items.

$$j = \text{floor}(U(i + 1)), \quad U \sim \text{Uniform}(0, 1) \quad (4)$$

Algorithm 1. Weak-skill-prioritized stage selection (documented rule)

Input: career, stage s , tagged question bank B , prior answered outcomes H

If $s > 1$, compute correctness ratio per tag with nonzero history

Order eligible candidates using the three lowest-ratio tags first

Select $Q = 5$ candidates subject to stage eligibility and availability

Shuffle selected questions and their option order using Fisher–Yates

Output: five tagged questions, transformed option maps, and answer keys.

D. Assessment Score, Focus Warnings, and Stage Gates

The assessment contains $Q = 5$ multiple-choice items per stage. A raw percentage R equals 100 times the number C of correct answers divided by Q , as in equation (5). The documented focus rule gives two grace warnings, then assesses a five-point penalty per additional warning: equation (6) defines chargeable warnings V and deductions D . The final rounded percentage S is equation (7), bounded below by zero. Progression occurs when the computed S is at least 60, as stated in equation (8).

The rule has substantial boundary implications. With exactly three correct responses, $R = 60$. At $W = 0, 1$, or 2 , $D = 0$ and $S = 60$, so the threshold is met. At $W = 3$, $D = 5$ and $S = 55$, so the same knowledge responses no longer pass. With four correct answers and $W = 6$, $S = 60$; with four correct answers and $W = 7$, $S = 55$. These are arithmetic consequences, not observed browser cases. The absolute score does not have an empirical accuracy interpretation.

Window blur and visibilitychange events can represent legitimate access needs such as assistive technology, browser permission prompts, or accidental notification focus changes. An anti-cheat policy based only on these events may penalize honest learners. The user interface should disclose the rule and allow accessible alternatives. It should also deduplicate events generated by one tab-away action; otherwise a single switch

might produce multiple warnings. This edge case requires event-trace reproduction.

$$R = 100 \frac{C}{Q}, \quad Q = 5 \quad (5)$$

$$V = \max(0, W - 2), \quad D = 5V \quad (6)$$

$$S = \text{round}[\max(0, R - D)] \quad (7)$$

$$P_{\text{stage}} = 1[S \geq 60] \quad (8)$$

TABLE IV. ANALYTICALLY DERIVED SCORE BOUNDARY CASES (NOT EXECUTED UI TESTS)

Correct / 5	Warnings W	Raw %	Final %	Gate
3	0	60	60	Pass
3	2	60	60	Pass
3	3	60	55	Fail
4	6	80	60	Pass
4	7	80	55	Fail
5	11	100	55	Fail

The three stages have different purposes: Stage 1 is Foundation, Stage 2 is Practical, and Stage 3 is Simulation. Later stages are unlocked only after passing earlier stages. The state machine must define whether a successful prior attempt remains passed after a later failed retry, and whether a reset affects past weak-skill statistics. The project description says stage locking and alternate questions are implemented, but it does not provide a test trace of every transition.

Five questions per stage make the raw score coarse: each additional correct item changes R by 20 percentage points before a warning deduction. Stage scores therefore should be interpreted as completion gates under specified rules rather than fine-grained estimates of a person's overall professional competence. Reliability and construct validity require many more calibrated items and independent evaluation.

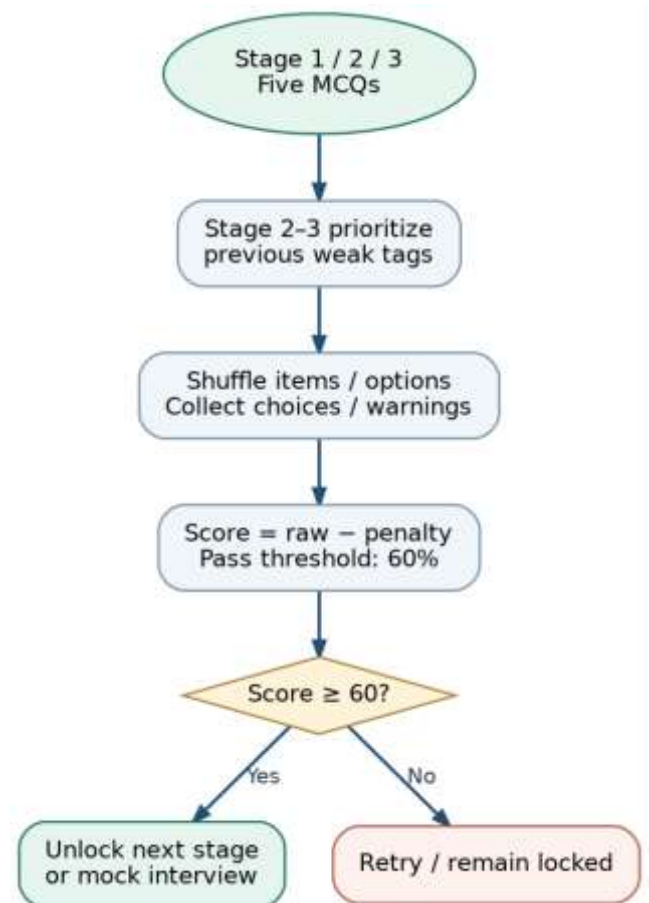


Fig. 4. Rule-based assessment workflow, weakness prioritization, score deduction, and progression branches.

E. Interview Answer Matching and Composite Score

After the learner passes all three assessment stages, the interview simulator reportedly administers six encounters: HR introduction, technical theory, code-output tracing, hands-on JavaScript execution, practical scenario, and closing. Text answers are normalized and tokenized; predefined concept aliases identify recognized phrases. For question q , let E_q be its nonempty set of expected concepts and M_q the subset recognized in the response. Equation (9) measures concept coverage, and equation (10) gives the reported rounded average across N scored questions.

This evaluation is reproducible if the concept dictionary and normalizer are fixed, but its semantics are limited. A correct paraphrase may fail matching if it uses an unlisted synonym; an irrelevant sentence may receive credit for mentioning isolated keywords. Negations, context, code syntax, and spoken punctuation can alter apparent meaning without changing simple token presence. Expert-annotated answers are needed to estimate false positives and false negatives. No such annotated reference corpus has been supplied.

The passing threshold for the overall interview is 60. The inventory reports that passing triggers `POST /api/course-results`, persisting a completed course mark in SQLite. The formula does not justify calling the score an employability probability; it summarizes coverage of the preselected concepts

in this particular scripted interview. The documentation does not provide per-role scoring rubrics, independent rater agreement, or interviewer calibration.

$$C_q = \frac{|M_q|}{|E_q|}, \quad |E_q| > 0 \quad (9)$$

$$I = \text{round} \left(\frac{100}{N} \sum_{q=1}^N C_q \right) \quad (10)$$

The code segment runs JavaScript snippets in a sandboxed iframe using restricted attributes and a Content Security Policy. It reportedly executes three input/output tests within a 1,500-ms timeout. The user-visible code runner is an execution evaluation component, not an AI model. Its quality depends on the appropriateness of hidden tests, errors surfaced to learners, and isolation between untrusted code and application data. Source inspection would be needed to establish whether the timeout also stops CPU-bound synchronous loops in the browser environment used.

Because roles include Business Analyst, UI/UX Designer, Data Analyst, Cloud Engineering, and QA, an exclusively JavaScript-focused coding challenge may not equally measure their job-specific competence. A future validation effort should ensure that each role's question blueprint and practical tasks reflect required skills, and that a passed JavaScript exercise is not presented as proof of readiness for a noncoding occupation.

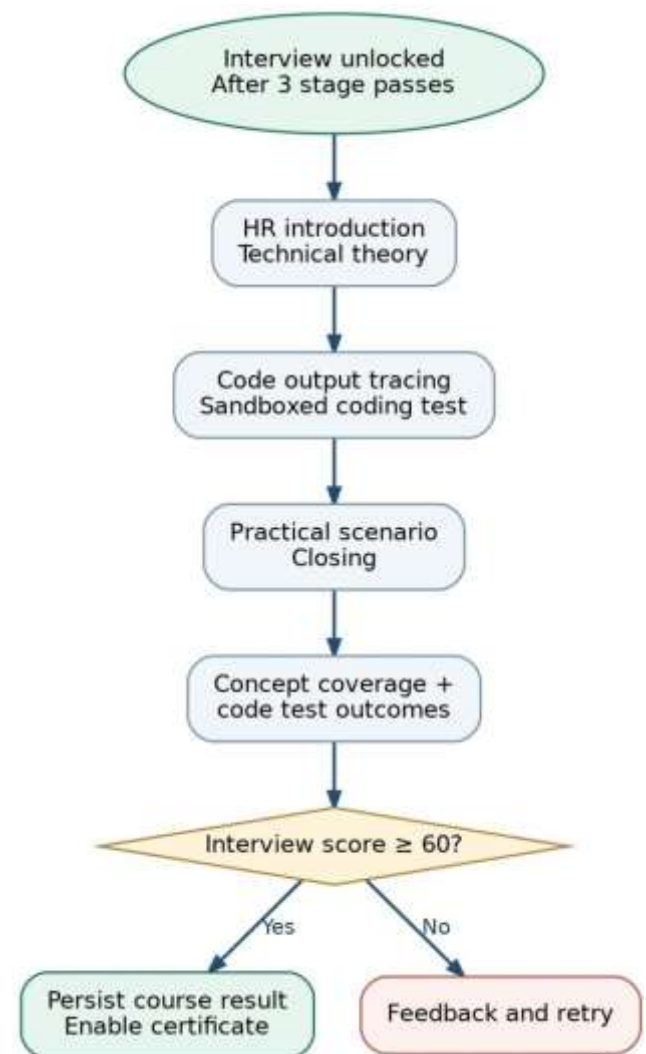


Fig. 5. Six documented interview phases, concept scoring, and protected result persistence after passing.

F. Readiness Categories and Skill Descriptives

The application reportedly aggregates tagged assessment outcomes into per-skill percent-correct values in equation (11). For an overall assessment summary, the mean of passed stage scores is calculated as equation (12), where J counts passed stages included in the average. This averaging rule conditions on passed stages and can therefore inflate the interpretation of performance if failed attempts are omitted; reports should disclose whether retries and failures contribute to a summary.

The documented readiness boundaries are below 40 = Beginner, 40 to less than 60 = Developing, 60 to less than 75 = Intermediate, 75 to less than 90 = Job Ready, and at least 90 = Advanced. These are hard-coded display labels, not externally validated cut scores. A learner with 74 and 75 receives distinct categories even if the score difference is less than the measurement error implied by a short assessment. The label 'Job Ready' is especially risky without role-specific occupational standards and employer evaluation.

A skill may have no answered questions; its percentage should then be treated as unavailable rather than zero. The

application should distinguish the absence of evidence from evidence of zero performance. The strongest and weakest categories should also include sample size so that one answered question does not appear as decisive as twenty responses.

$$S_k = 100 \frac{c_k}{n_k}, \quad k \in \mathcal{K} \quad (11)$$

$$S^- = \text{round} \left(\frac{1}{J} \sum_{j=1}^J S_j \right) \quad (12)$$

TABLE V. DOCUMENTED READINESS-TIER MAPPING

Assessment aggregate	Interface descriptor	Interpretation constraint
0–39	Beginner	Internal score band only
40–59	Developing	Internal score band only
60–74	Intermediate	Threshold-based application label
75–89	Job Ready	Not employer-certified readiness
90–100	Advanced	Not calibrated professional proficiency

G. Certificate Identifier and Evidence of Issuance

The Digital Skill Passport becomes available after all three assessments and the mock interview meet their passing requirements. The documented identifier combines learner email, selected career, and a salt string, then applies a 32-bit Fowler–Noll–Vo FNV-1a hash. With $h_0 = 2166136261$, each character byte b_t updates the running hash using equation (13); the base-36 token is assembled into an identifier containing a career marker and year. Stable inputs produce repeatable IDs.

FNV-1a is designed for fast noncryptographic hashing, not credential authenticity. A determined attacker may create a similar-looking string or exploit collisions in a limited 32-bit space. A user-facing PDF generated by jsPDF can be copied or edited unless a separate server-side verification or signature system exists. The present evidence supports a printable completion document and deterministic label, not an externally trustworthy proof that an employer can verify online.

The W3C verifiable-credential framework [11] illustrates requirements for a future issuer-backed implementation:

VI. DETAILED IMPLEMENTATION AND USER WORKFLOWS

A. Account Registration, Login, and Session Recovery

The first user interaction is a public landing or account page. Registration accepts a display name, email, password, and confirmation; the server checks email shape and password bounds, derives a scrypt password hash using a 16-byte random salt, and inserts a unique user record. A successful registration establishes an authenticated session and supports entry to the learner dashboard. A repeat registration with the same case-insensitive email should fail predictably and should not overwrite the original credential. The documented backend integration tests include registration isolation and hashing behavior, but the exact observed responses are not included in the uploaded inventory.

On login, the backend compares the submitted password against its stored hash using a timing-safe approach. A successful attempt updates recent-login state and emits audit information; an unsuccessful attempt should not reveal whether a user exists. Session identity then travels through a signed HttpOnly cookie. AuthContext exposes the sanitized name and email to protected screens, while Express derives the user ID from the cookie for protected course operations. This is a safer ownership pattern than trusting a user_id in an editable browser request body.

Profile management permits the display name to change through a protected PATCH operation. Its stored name may affect the printed certificate; therefore, certificate names generated after a profile update should reflect authenticated profile data. The recovery pages are explicitly described as a demo reset: they check the submitted email and replace the stored password hash without external email or one-time-password delivery. Such a demonstration must not be mistaken for a production-safe account-recovery mechanism, as knowledge of an email address alone is not possession proof.

unambiguous issuer identity, signed claims, revocation or status mechanisms, secure key management, and privacy-aware disclosure. Until those are implemented and tested, marketing and manuscript text should avoid equating CareerAI's passport with a digitally signed professional license.

$$h_{t+1} = [(h_t \oplus b_t) \cdot 16777619] \bmod 2^{32} \quad (13)$$

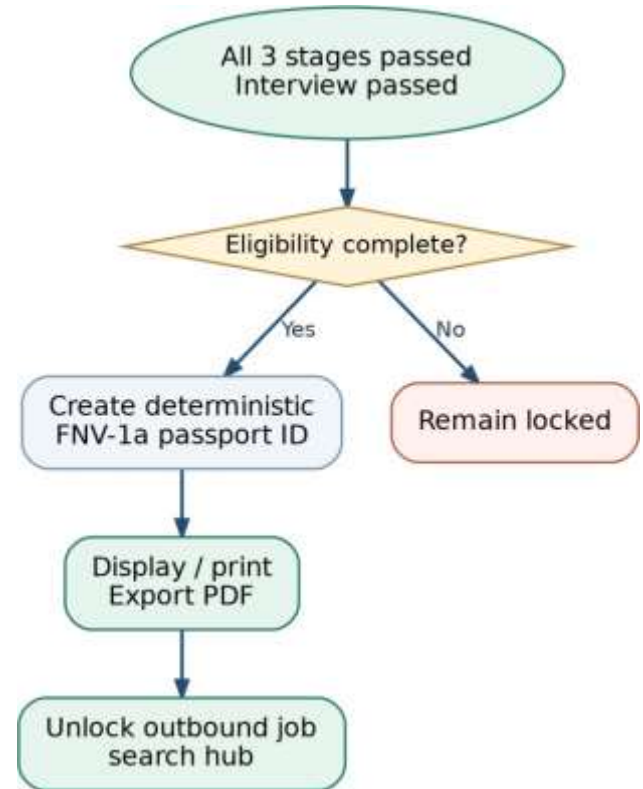


Fig. 6. Reported certificate eligibility, deterministic identifier generation, PDF export, and job-link unlocking.



The inventory includes migration from older browser-only plaintext accounts. The frontend should detect a legacy record only during a controlled migration path, post credentials through the protected channel, and avoid exposing legacy secrets in logs or UI. Unresolved conflicts and interrupted migration represent important tests. Since the live code is not available, this manuscript cannot establish that the migration removes every plaintext copy or that error conditions are fully handled.

B. Learner Dashboard and Role Selection

The dashboard coordinates the learning journey rather than performing assessment itself. It supplies an entry to the role catalog, shows progress or readiness summaries, and routes users toward learning, tests, interviews, certificate, and jobs. Navigation visibility and authorization are not the same: a hidden button is a user-experience aid, whereas server-protected operations and reliable state checks enforce business rules. The browser should not be able to issue a course mark merely by making a hidden page visible.

The catalog defines eight supported career pathways: Java Full Stack Developer, Cloud Engineering, DevOps, Data Analyst, Data Scientist, Business Analyst, QA / Software Tester, and UI/UX Designer. The supplied data description states that each has eight curricular topics, giving 64 topic entries in total. The actual names of all 64 topics are not reproduced because the full careers.js data file was not provided; inventing topic titles would create false evidence of curricular breadth. The paper instead records the reported cardinality, role identities, and catalog behavior.

Selecting a career establishes a role context for the subsequent syllabus, assessment bank, interview scenarios, and job-search query. When switching roles, the client reportedly resets some in-memory assessment and interview states but preserves per-email stored progress. Such a design raises a consistency question: completed progress should be scoped by both user and career, not only by user. A recommended verification is to pass a stage for one role, switch to another, and confirm that the new role does not inherit an irrelevant credential gate.

Workflow evidence — Dashboard after a completed assessment journey

In the user-supplied sample state (Fig. 7), the dashboard shows 13% topic progress, three of three stages passed, an interview score of 84%, and certificate eligibility. The different denominators matter: assessment readiness is not equivalent to curriculum completion. These are values displayed for a particular demonstration account, not aggregate effectiveness statistics.

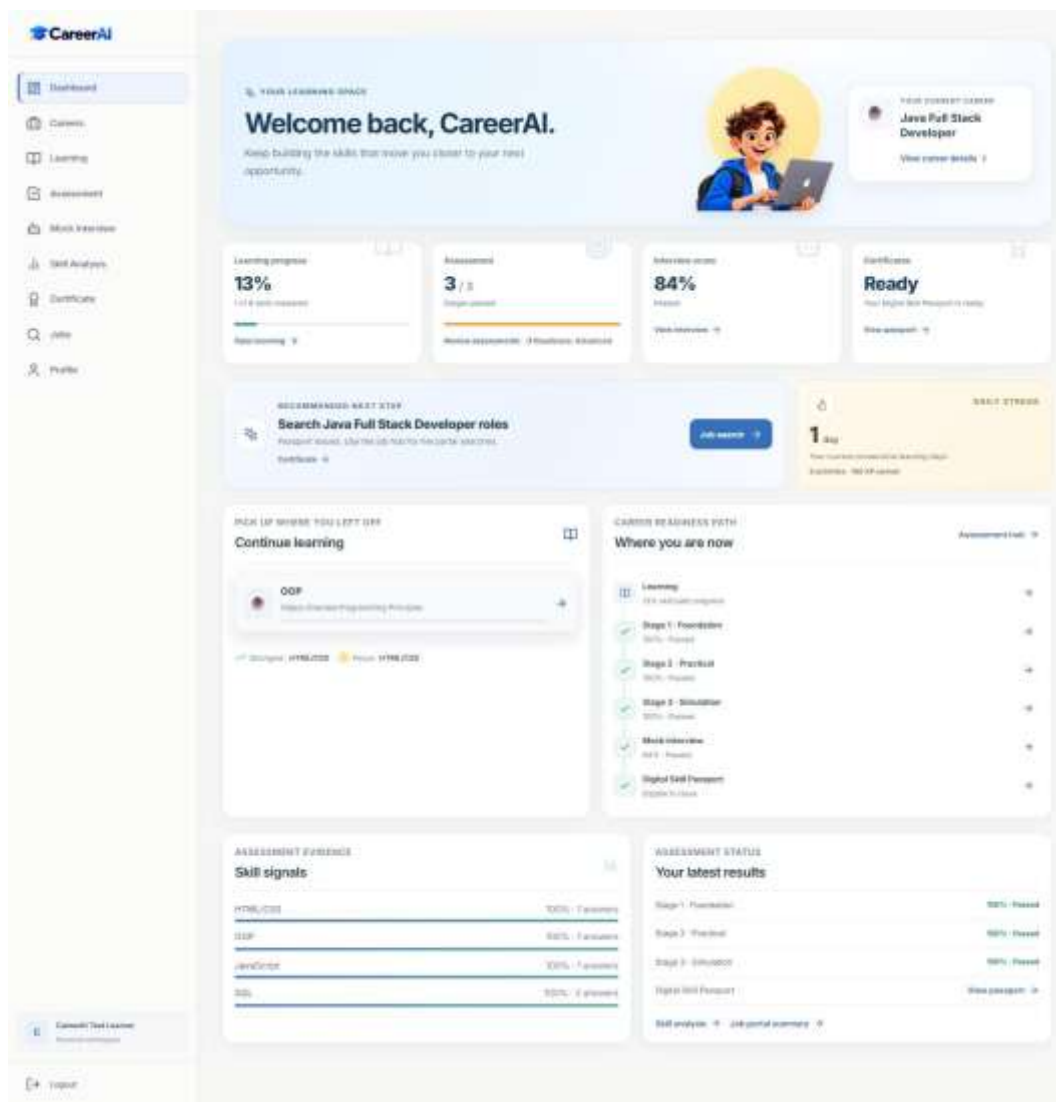


Fig. 7. User-supplied learner dashboard showing 13% learning progress, three passed assessment stages, 84% interview score, and passport readiness.

TABLE VI. SUPPORTED CAREER CATEGORIES (DOCUMENTED IN CAREERS.JS INVENTORY)

Pathway	Role orientation	Evaluation implication
Java Full Stack Developer	Application programming	Role-specific code + architecture items
Cloud Engineering	Infrastructure and services	Deployment / cloud fundamentals
DevOps	Delivery and operations	Automation, CI/CD and reliability
Data Analyst	Data interpretation	SQL / analysis reasoning
Data Scientist	Statistical modeling	Statistics and model foundations
Business Analyst	Requirements and processes	Stakeholder / process scenarios
QA / Software Tester	Verification and quality	Test design and defect reasoning
UI/UX Designer	Design and usability	Research / interaction critique

The middle column of Table VI is a general role orientation, not a claim that a particular undocumented question exists in CareerAI. The evaluation column states the kind of role alignment that future content validation should check. A promotional AI/ML Engineer card is reportedly visible on the landing page but is not defined in the eight supported careers; it is therefore excluded from the implemented-role count. This difference is useful evidence of why UI claims need to be reconciled with canonical data sources.

Workflow evidence — Career pathway catalog

Figure 8 shows the eight implemented pathways as selectable cards, including Java Full Stack Developer, Cloud Engineering, DevOps, Data Analyst, Data Scientist, Business Analyst, QA/Software Tester, and UI/UX Designer. This observed catalog should take precedence over promotional landing-page tiles that may describe unsupported tracks.

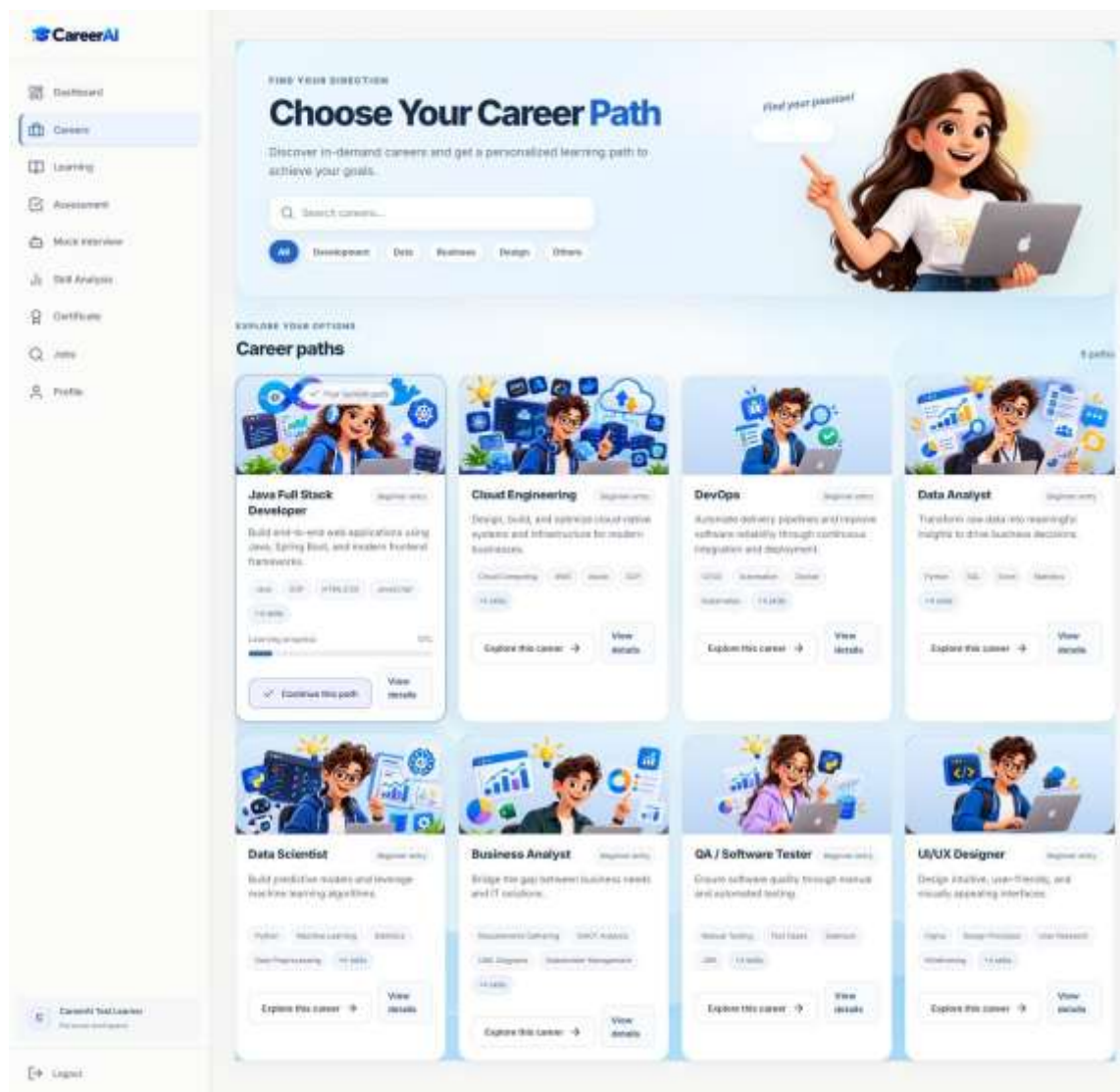


Fig. 8. User-supplied eight-path career catalog with role-specific selection controls and progress context.

C. Curriculum Catalog and Roadmap Navigation

The learning page presents role-dependent topic cards. Clicking a topic opens a roadmap route parameterized by a topic ID. The roadmap supplies curated video tutorials, reading links, concepts to review, a practical exercise, and a mini-project. These resources are assembled from static JavaScript data files rather than queried from a trained recommendation system. The route slug resolver and topic normalizer are reportedly covered in learning tests. A broken slug should produce a controlled error or return to a valid catalog, not a blank page.

The first Learn phase uses external reading and video destinations. Recording a video click is weaker evidence than confirming that a learner watched the material or comprehended it. The system reportedly awards the phase XP after a completion interaction, but no video progress tracking, playback verification, or externally graded content assessment is documented. A fair methodology description therefore says that the phase records self-directed resource completion, not verified instructional exposure.

In Try, the learner solves a mini-exercise and submits a reflection satisfying the documented minimum of 16 characters. Draft text is auto-saved locally. A character threshold checks presence of writing, not quality of reasoning. Entries such as meaningless repeated characters may satisfy the length guard unless semantic validation is added. A corresponding test must distinguish empty, whitespace-only, below-threshold, and valid-content submissions. The exact treatment of leading and trailing spaces is a code-level question that cannot be settled from the inventory alone.

Build encourages an applied mini-project and then records confirmation and reflection. No independent project artifact review, repository inspection, plagiarism check, or mentor scoring is described. Its Mastered status should thus be labeled a completion state, not certified skill mastery. The same caution applies to XP and daily streaks: they are engagement artifacts generated under predetermined rules and should not be used as a proxy for externally established learning outcomes.

Workflow evidence — Object-oriented programming learning resources

Figure 9 captures the Object-Oriented Programming roadmap before phase completion. It shows the Learn phase available, the Try and Build phases locked, plus curated video and reading entries. The visible zero-progress state illustrates the declared prerequisite mechanism; actual link health and resource quality require independent checking.

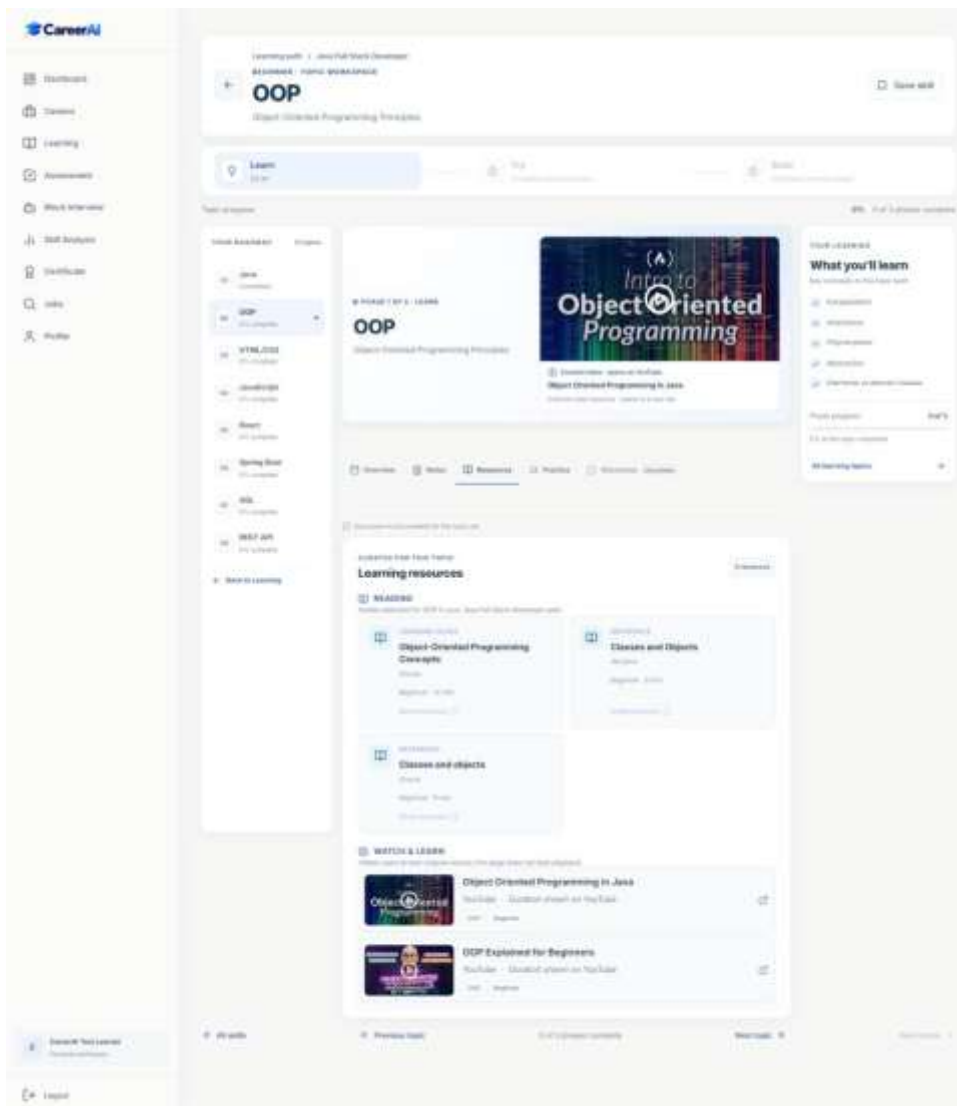


Fig. 9. User-supplied OOP roadmap at the Learn stage, with locked subsequent phases and curated video/reading resources.

D. Assessment Hub, Attempt Lifecycle, and Retrying

The assessment hub displays three stages and their respective lock states. When a learner starts a stage, the application chooses five appropriate career-specific questions, shuffles option positions, and records the attempt. Stage 1 covers foundations; later stages prioritize skills with lower historical success rates. The question banks are reported in `questions.js` and `questionsExtra.js`. The document states that the data sources include at least five questions per stage for each career, a condition suitable for direct automated verification of all supported roles.

On each response, the client records the selected option and updates the attempt. The runner blocks several browser events—copy, cut, paste, and contextmenu—and listens for loss of visibility and focus. A warning counter increases under the reported policy. At submission, correct answers determine raw marks, extra warnings determine deductions, and the final value determines pass and next-stage unlock. The score display should distinguish the raw correct count from the penalty so that a learner knows why five correctly answered questions could nonetheless receive fewer than 100 marks.

A retry should use a fresh question presentation and avoid carrying selected options from the failed attempt. The document says question order and answer order change across attempts and missed skills receive priority, but it does not show a full state-transition log. For validity, test cases must cover immediate retry, revisit after refresh, stage navigation by browser history, and career changes.

A user who directly enters a Stage 3 URL should be redirected or denied if Stage 2 has not passed, independently of whether a visual lock icon was shown.

The use of fixed passing thresholds across highly different roles is an engineering convenience. A common 60-point threshold does not imply common item difficulty or equivalent competencies. Moreover, five items yield low score resolution and increased exposure risk across retries; future empirical work must investigate alternate forms, item difficulty, discriminative value, and stability of scores before using stage completion to make career-readiness claims.

Workflow evidence — Simulation-stage question interface

Figure 10 illustrates stage 3 of 3 with question five about the React side-effect hook and four of five answers already recorded. The screenshot substantiates question presentation, progress tracking, and role labels. It does not, by itself, demonstrate the correctness of adaptive selection, enforcement of tab-switch penalties, or a submitted stage score.

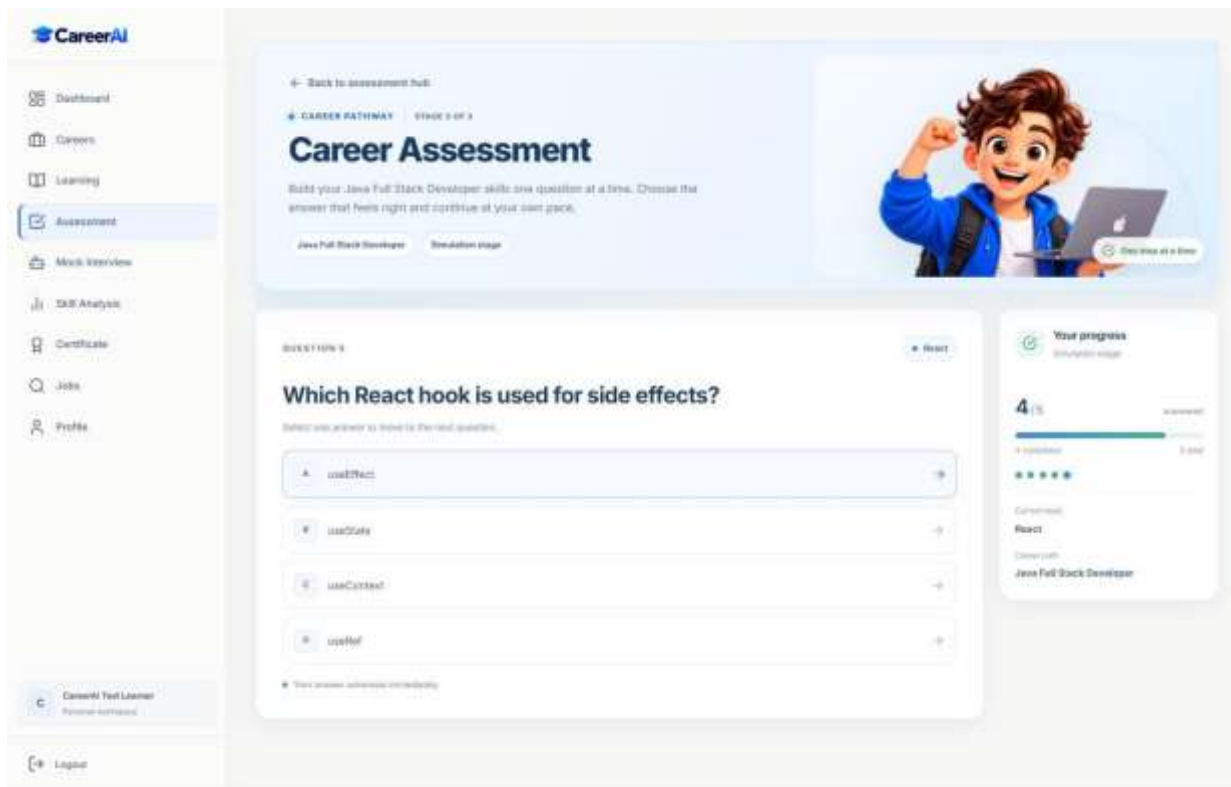


Fig. 10. User-supplied assessment interface at Stage 3, question 5, showing a React multiple-choice item and four answers recorded.

E. Skill Analysis and Personalized Review

The skill analysis page derives percent-correct summaries from tagged historical assessment responses and interview observations. The reported outputs include the learner's strongest and weakest skill tags, a readiness descriptor, score bars, and links to relevant revision resources. The recommendations are curated and selected using the identified weak area; no model training, embedding retrieval, or semantic inference service is indicated in the implementation inventory.

An honest analysis panel should reveal the denominator underlying each skill percentage and should avoid overinterpreting a small sample. A skill with one wrong answer has 0% observed correctness but uncertain underlying competency; one with ten wrong responses is more informative. These distinctions matter when recommendations are sorted. The app's use of a simple observed ratio supports quick deterministic feedback, but meaningful remediation requires reviewing question coverage, tag validity, and whether resource links actually address the diagnosed gap.

No classroom or industry benchmark is identified for the readiness thresholds. For reporting purposes, the categories should be explicitly described as application-defined. A possible user-facing improvement is to show 'assessment readiness within CareerAI' rather than 'job ready' unless employer-rater studies support the latter. This is an ethical and validity distinction, not a criticism of having a progress-oriented dashboard.

Workflow evidence — Skill breakdown and readiness categorization

Figure 11 displays 100% for several topic tags, with small item counts—one HTML/CSS item and one OOP item, for example. A perfect mark on a single item is an imprecise estimate of mastery. The interface labels readiness as Advanced, but no external readiness criterion or industry competency examination is supplied.

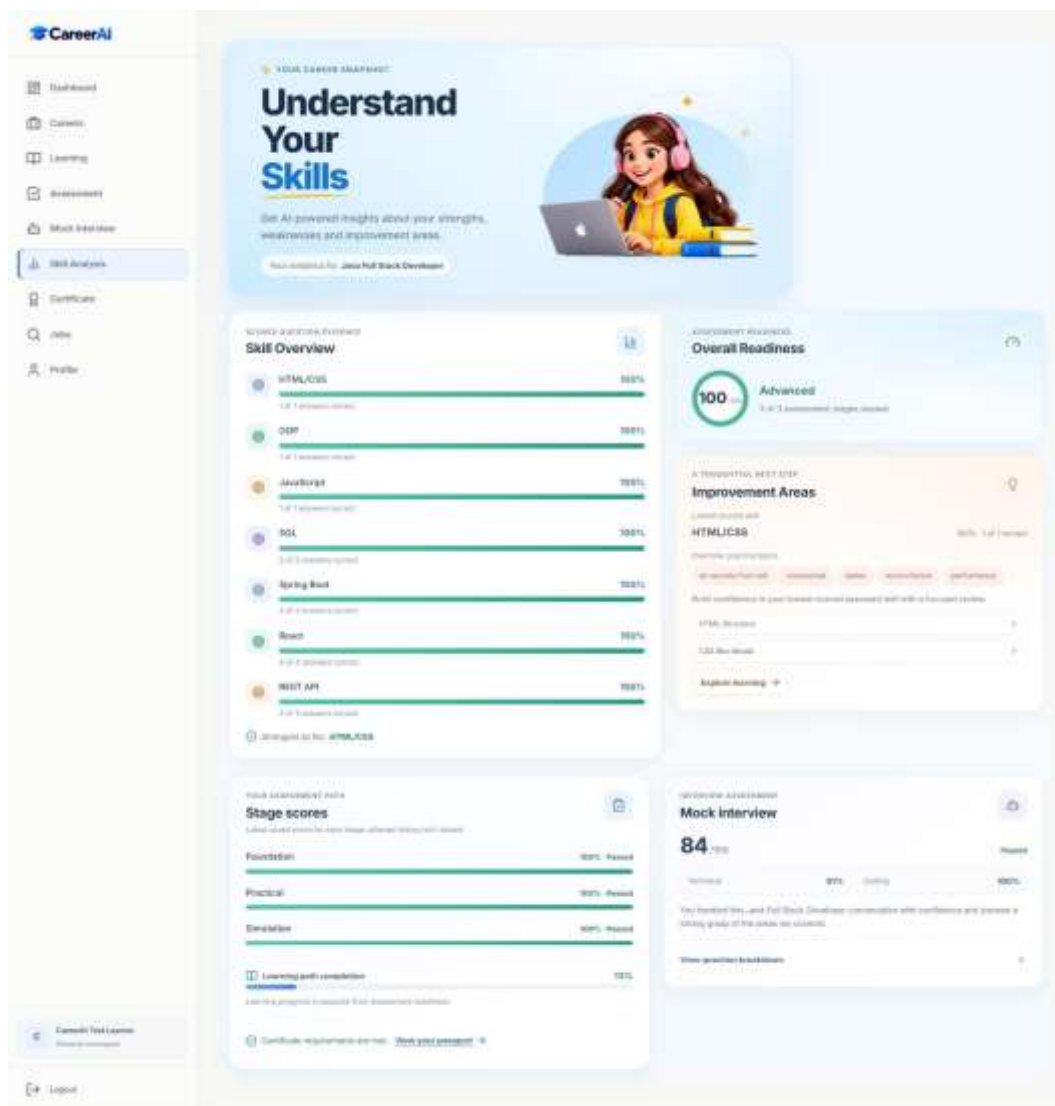


Fig. 11. User-supplied skill analysis screen showing topic-level percentages, denominators, stage marks, interview score, and rule-based readiness.

F. Mock Interview, Speech Interfaces, and Code Runner

The mock interview opens only after successful completion of the three assessment stages. It resembles a scripted interviewer that asks an introductory question, proceeds to theory, displays an output-tracing task, provides a JavaScript coding exercise, asks a scenario question, and closes with feedback. Browser SpeechRecognition or webkitSpeechRecognition supplies optional voice-to-text, and speechSynthesis provides synthesized interviewer audio. The transcription engine is a browser service rather than an in-house trained model; its language support and runtime availability depend on client configuration.

For concept-matched answers, the scoring function normalizes the text and searches for expected tokens or recognized aliases. Feedback may report covered concepts and missed concepts. A rigorous experiment should compare those outputs to a blind panel of assessors, especially for valid answers using unanticipated wording and incorrect answers that mention all expected keywords. The interface should not claim to understand the full meaning of a response when the implementation uses token matching.

The coding segment executes input text as JavaScript in a sandboxed iframe under a restrictive policy. The inventory specifies sandbox=allow-scripts and a CSP denying network connections and most other resources. It also specifies three expected test cases and a 1500-ms timeout. These are useful design controls, but the code-runner's termination and cross-origin isolation must be verified in the actual supported browsers. Particularly important adversarial tests include asynchronous timers, infinite loops, attempts to reference parent state, and very large generated output.

When an interview score meets the reported 60 threshold, the client posts a course mark to SQLite. The profile and admin dashboard can then report the completed mark. The exact division between scores calculated in the browser and marks accepted by the server is a credibility boundary: a trusted certificate system should retain the question set, rubric version, and validated responses on the server or recompute marks there. The uploaded text does not establish that stronger property.

Workflow evidence — Interview scenario and coding feedback

Figure 12 shows a scenario question about stale data after saving and a draft response in the text area; earlier dialogue reports “3 of 3 tests passed” for a coding task. That test count belongs to the in-browser coding challenge for this attempt, not the 19-case repository regression suite. The screenshot is evidence of interface feedback, not an independently audited interview mark.

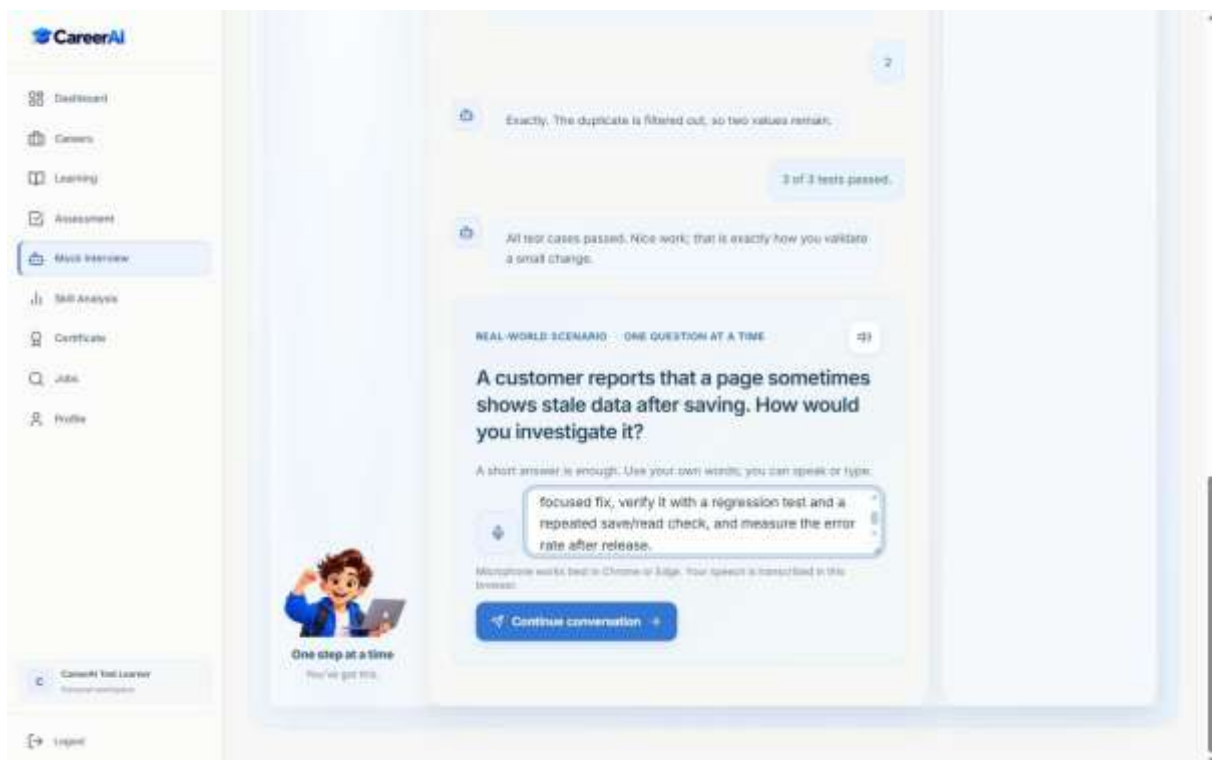


Fig. 12. User-supplied mock interview conversation, coding-test feedback, and a scenario response entry.

G. Digital Skill Passport and Print Workflow

The certificate page checks stage-completion and interview gates. For eligible users it presents a named completion document, generates the stable identifier described earlier, and offers landscape PDF download through jsPDF or a browser print path controlled by CSS. Both are user-facing export functions. They can convey an application-completion record but cannot independently attest to occupational competency or authenticity if any client can reproduce the identifier and layout.

A verification process should compare repeated exports for the same learner/role/year, changed profile names, career switches, and expired or cleared sessions. Filename sanitation and handling of unusual Unicode names also need tests. An employer receiving a PDF cannot be assumed to access the original account database, so verification would need a public issuer-signed record or status endpoint and a policy for corrections and revocations. These facilities are not documented as implemented and are therefore future work.

The supplied certificate screenshot is incorporated as Fig. 13 and depicts an issued example display. An independent test would still need to invoke download/print, compare the exported file with the on-screen mark, and test the identifier against the underlying issuer record. The visual screenshot alone is not proof of cryptographic verification.

Workflow evidence — Digital Skill Passport display

The supplied certificate interface (Fig. 13) presents a sample CareerAI Test Learner certificate, Java Full Stack Developer role, 100% assessment summary, 84% interview mark, and a deterministic-looking credential identifier. It documents an issued display state, but there is no evidence of third-party credential validation, PDF export execution, or cryptographic signature verification.

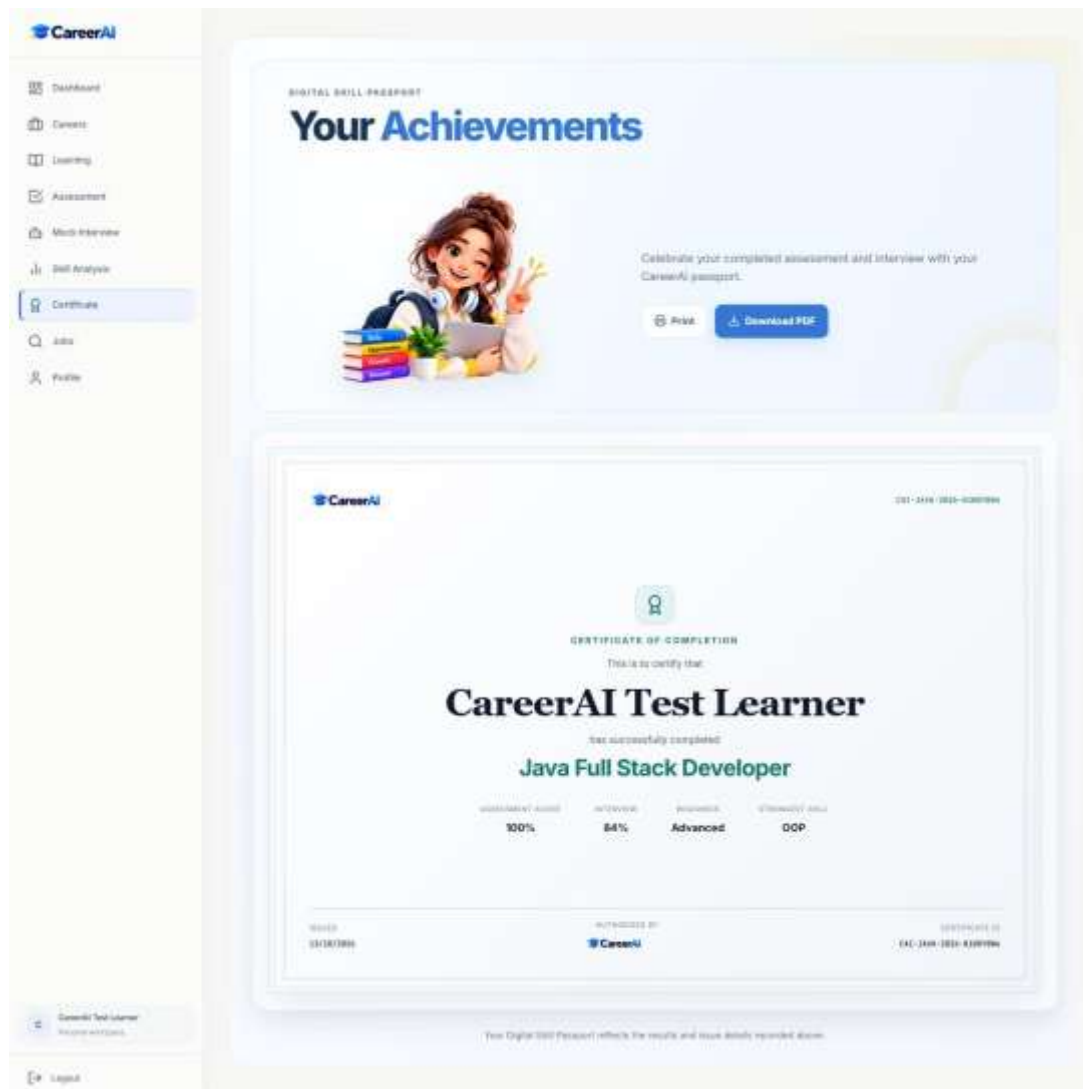


Fig. 13. User-supplied unlocked Digital Skill Passport with sample learner, career, displayed scores, and print/download controls.

H. Job Search Hub and External Data Limitations

After passport issuance, CareerAI reportedly exposes outbound search links for LinkedIn, Indeed, Naukri, and Google Jobs. The user can enter a location, with Chennai as a documented default, and the selected role becomes part of a URL-encoded query. The browser then opens or navigates to the corresponding external destination. This is an external search-link integration; it is not a live vacancy aggregation API, company verification engine, application tracker, or salary prediction model.

Because results are rendered and ranked by third-party services, CareerAI cannot claim ranking quality or vacancy completeness unless it separately captures and assesses the returned listings. Search URLs may become invalid as external websites change. The testable responsibilities of CareerAI are to use correct query encoding, include the intended career and location, open the requested target safely, and avoid exposing sensitive internal data in URLs. A future integrated jobs product would need employer-source provenance, duplication handling, expiry checks, and an evaluation corpus.

Workflow evidence — External jobs search with location entry

In Fig. 14, the learner has specified Remote as the location, and the interface offers outbound search actions for LinkedIn, Indeed, Naukri, and Google Jobs. CareerAI is not shown retrieving or ranking live vacancies itself. Employment listing freshness and final applications remain controlled by the external destination.

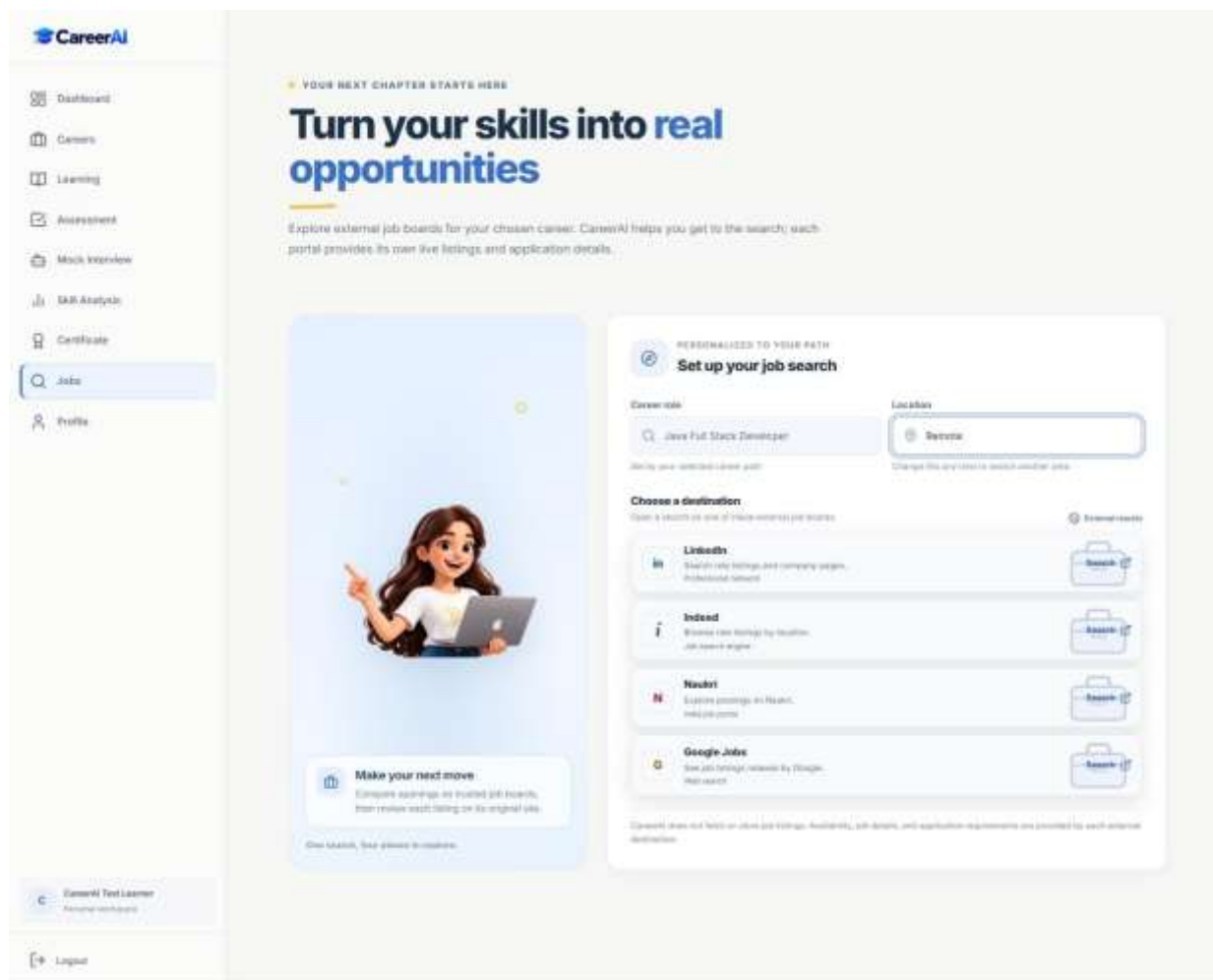


Fig. 14. User-supplied jobs hub with the Remote location filter and four outbound job-board search destinations.

I. Administrator Portal and Audit Functions

The owner portal has its own login page and session cookie, `careerai_admin_session`. Its dashboard reportedly aggregates total registered learners, successful logins in the prior seven days, and completed courses. Administrators can search paginated user lists, inspect per-user course marks and activity, and review login history. The backend implements distinct `/api/admin/*` routes with owner authorization checks, protecting personal usage information from ordinary users.

The audit system records `account_created`, `login_success`, `logout`, `profile_updated`, and other named activities in the appropriate tables. An audit trail is only as useful as its completeness and integrity. The inventory does not provide a tamper-evident logging policy, data-retention schedule, export approval process, or scope of the administrator's read access. The ability to inspect usage patterns should therefore be balanced against data-minimization and privacy expectations.

Test design should include a learner attempting direct GET `/api/admin/users` requests, an owner session expiring during pagination, an unknown user ID, escaped search input, and concurrent account creation while browsing paginated lists. A frontend route guard alone is insufficient to satisfy the authorization requirement. The reported backend integration suite includes owner authorization, but the raw requests and expected status codes were not provided.

Workflow evidence — Owner administrative dashboard

Figure 15 documents an owner view reporting two registered users, two recent user logins, one completed course, and one result, with a user table and activity timeline. These are snapshot counts from the supplied image, not system-wide statistics or evidence of a live production deployment. Personal account identifiers have been obscured for publication; metrics and event classes are preserved.

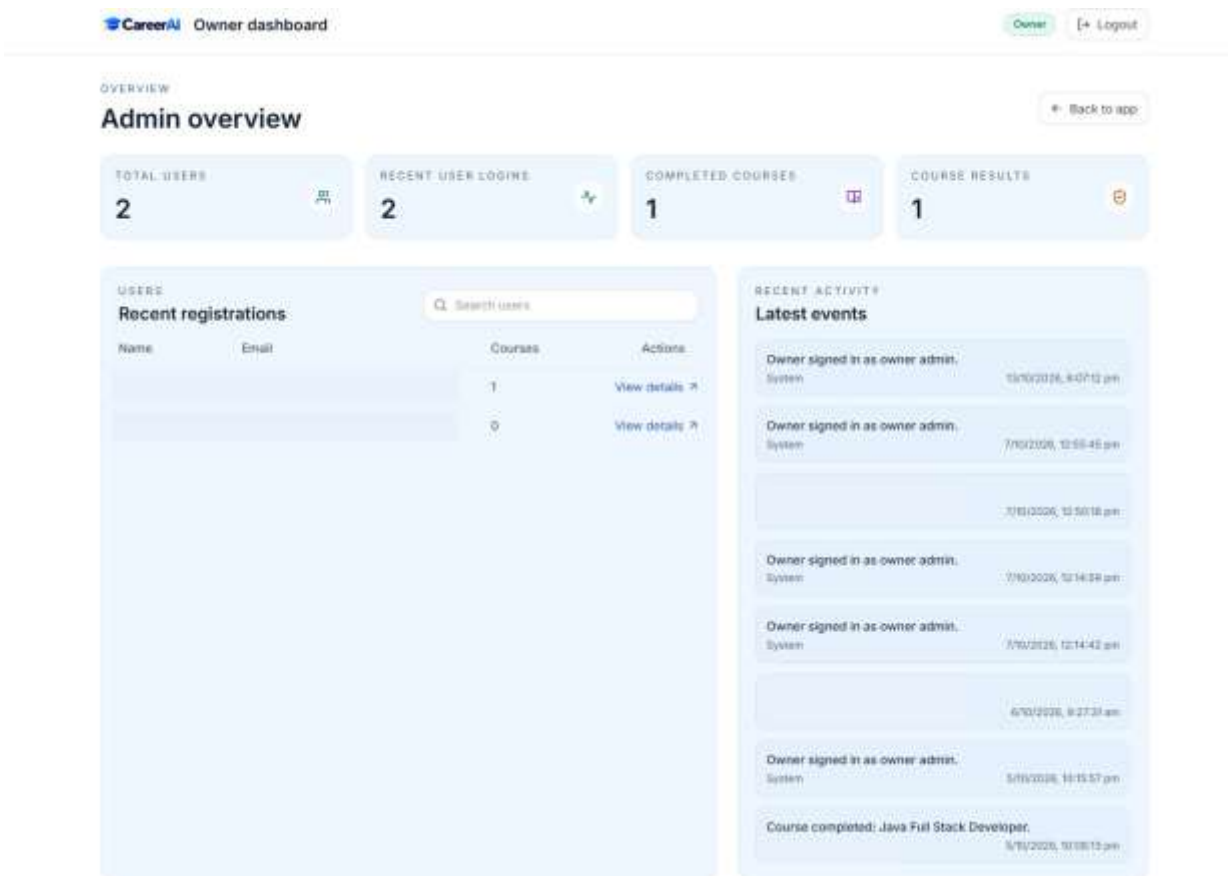


Fig. 15. User-supplied owner dashboard showing snapshot aggregate metrics, user directory, and recent audit events; personal identifiers redacted.

VII. END-TO-END CONTROL FLOW AND STATE TRANSITIONS

A. Ordinary Learner Journey

Fig. 16 depicts the supported conceptual journey. Starting from registration, the user selects a career, works through resources and activities, completes ordered stage assessments, undertakes the mock interview, receives a passport if eligible, and accesses external job searches. The path should be understood as gated but not strictly linear in time: a learner may revisit materials, compare topics, retry a failed test, and review analytics before or after a later stage. The route hierarchy itself does not prove that every gate is enforced securely.

The most important operational event is a transition from noneligible to eligible. At each gate the application should compute eligibility from documented state, persist a record, and re-evaluate after a reload. A failed attempt should not cause unrelated progress to disappear, and the system should not allow an expired or different account to reuse the previous learner's browser-local success flags. This cross-account condition is particularly important when a shared computer is used for a demonstration.

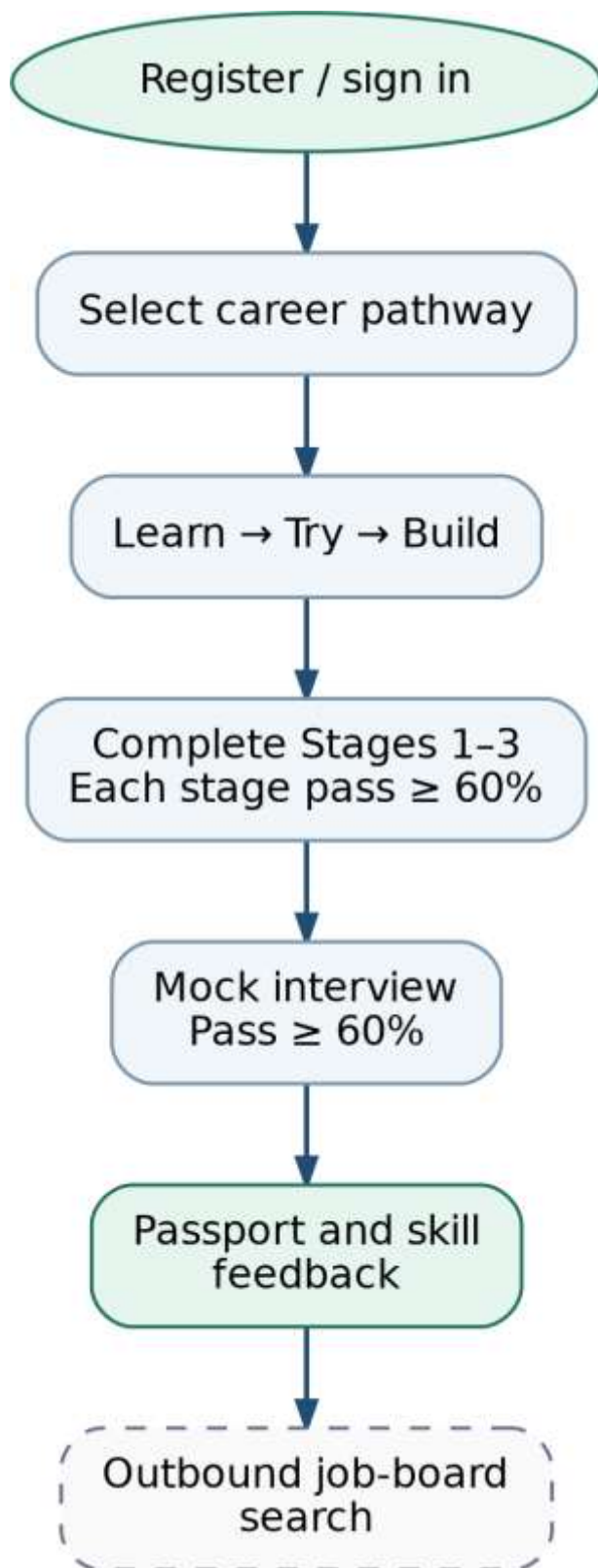


Fig. 16. Documented learner sequence and score-related branches from registration through job-board search.

B. Assessment State Machine and Retry Invariants

The assessment machine can be analyzed as an ordered sequence of locked, available, attempted, passed, and failed states. Stage 1 begins available. Passing Stage 1 makes Stage 2 available; passing Stage 2 makes Stage 3 available. A failed stage should remain available for another attempt while the later stages stay locked. Because the question selector uses history, retrying should update weakness estimates only after the attempt is finalized and stored, not after each provisional click.

The key invariants are monotonic access after a recorded pass, lack of privilege escalation through URL guessing, correctness of option mapping after shuffling, and consistent score deduction for a warning event. The documentation reports tests for penalty and adaptive selection, but not browser-level proof of every invariant. A complete execution record should contain an attempt ID, stage, question IDs, selected answer IDs, warning count, raw marks, final marks, and gate decision. The inventory does not claim that all these fields are stored server-side.

C. Interview-to-Certificate Data Sequence

Interview completion crosses a different boundary from a learning checkbox. The client receives the recognized answer concepts and an aggregate interview score, and on pass invokes a protected course-results write. The server should validate the learner identity and write one mark for the relevant course; a later profile read should retrieve the same mark. A certificate download then renders user-facing metadata and the deterministic ID. An invalid or stale cookie must not write a mark for a different user.

The documented implementation has a course-results `UNIQUE(user_id, course_id)` constraint with update-on-conflict behavior. That supports a single latest mark per learner/course but does not preserve all historical submissions in the course_results table. If research questions require repeated-attempt analysis or longitudinal improvement, a separate immutable attempts table should be created with timestamps, scoring version, and question set references. The present inventory does not establish such durable fine-grained retention.

D. Owner Audit and Operational Traceability

The owner workflow is distinct: authenticate as owner, fetch dashboard aggregates, search a learner, view their audit history, and inspect course results. Every query should be performed through server-authorized endpoints and scoped by an explicit data-governance policy. Aggregated counts should have documented definitions: 'completed course' could mean one record, one passing interview, or one full certificate, and those are not necessarily the same quantity unless the server schema and insert rules enforce equivalence.

The end-to-end traceability goal is that a course completion can be related to the learner and the rule that generated it. The current design can report course marks but lacks evidence of end-to-end immutable attempt logs, human review, and credential status. A future validation event should compare the browser success state, the protected profile endpoint, the

SQLite record, and the owner display for the same authorized fixture without exposing live learner information.

VIII. EVALUATION METHODOLOGY, EVIDENCE AND ANALYTIC RESULTS

A. Evidence Classes and Data Provenance

The available evidence combines the project implementation inventory with nine retained user-supplied screenshots showing public, learner, and administrative interfaces. The inventory names versions, endpoints, formulas, and an earlier test run, but is not equivalent to reviewing all source files at a fixed commit. No complete version-controlled source tree, dependency lockfile, current independent browser test session, SQLite database export, network trace, or repository test log was provided here. The screenshots document rendered states and sample values but lack a complete capture manifest and matching source commit; no image is represented as a newly executed test by the manuscript editor.

This paper uses explicit evidence classes. D means described in the supplied project inventory; R means an execution outcome reported in that inventory but not independently replayed; A means a mathematical consequence derived from stated algorithms and independently recalculable without running the UI; P means a proposed future test; and N means evidence unavailable. Published research and standards are cited separately with numbered references. None of the A-class boundary calculations is labeled as a measured system result.

The described test run totals 19 passed tests out of 19 executed, with approximately 14.08 seconds elapsed. Four are identified as backend integration tests, five as assessment regressions, and ten as learning-logic checks. Because the program and test log were not available to rerun, the statement is correctly phrased 'the supplied documentation reports'. A reader should not infer zero defects across the system from zero observed failures in an unexamined selected suite.

Visual evidence consists of nine retained user-supplied screenshots (Figs. 7–15). They are treated as captures of interface states. One-screen percentage scores and owner counts lack study denominators, sampling protocols, and execution logs; they must not be treated as population-level outcomes. Fig. 16 remains a documentation-derived end-to-end schematic and Fig. 17 distinguishes documented claims, supplied visual evidence, and independent reproduction status.

TABLE VII. EVIDENCE CLASSIFICATION AND INTERPRETIVE SCOPE

Class	Available basis	Permissible conclusion
D	Uploaded implementation inventory	Documented architecture or claimed feature
R	Inventory statement of a previous test run	Reported test result, not independently reproduced
A	Arithmetic from explicitly given rules	Analytical boundary expectation
P	Specified test plan without run logs	Experiment proposed, not completed
N	No authentic evidence available	Do not report as result

B. Reproducible Local Setup and Version Control

A genuine replication should start by preserving a versioned snapshot of the ACUI_careerAI directory. Record the commit hash, uncommitted changes, Node and npm versions, operating system, browser name and version, Vite, Express, SQLite, and dependency lockfile hash. The inventory reports React ^18.2.0, React Router ^7.18.4, Vite ^5.1.4, Express ^5.2.1, better-sqlite3 ^13.0.3, and jsPDF ^4.2.1. The caret constraints are ranges, not guaranteed installed patch versions. A recorded npm install alone is insufficient to identify the exact packages used unless the lockfile is kept.

A minimal local startup uses npm install, npm run server, and npm run dev, with development API port 3001 and browser frontend port 4173. The .env must contain owner credentials and must not be committed. Before running, create a disposable SQLite fixture rather than overwriting production or personal learning records. Verify GET /api/health and basic registration with synthetic data. The repo is required to establish whether any seed data, migrations, or scripts have changed since the reported test outcome.

Browser reproduction should use a fresh profile for each independent account, then repeat critical flows after reload and in a second browser. Review the resulting SQLite database state with foreign keys enabled. Preserve console messages, HTTP status codes, time-stamped screenshots, test outputs, and explicit expected/observed pairs. The actual study reported here could not complete this setup because the executable project source was not available as part of the materials being evaluated.

C. Functional Test Matrix for Learner Identity

TABLE VIII. IDENTITY AND AUTHORIZATION TEST DESIGN (ALL P UNLESS RAW LOGS EXIST)

Case	Stimulus	Expected observable result
ID-01	Register valid synthetic account	User row + authenticated cookie
ID-02	Register duplicate email (case changed)	Reject duplicate; preserve first account
ID-03	Sign in with wrong password	Deny; no protected profile
ID-04	Call profile without cookie	Authorization failure
ID-05	Send another user ID in mark body	Server uses cookie identity, not body ID
ID-06	Learner calls /api/admin/users	Owner authorization denial
ID-07	Logout; reuse old cookie	Session rejected per effective logout policy
ID-08	Demo reset using unknown email	Controlled response; no account changed

These tests differentiate identity correctness from interface correctness. It is possible for the login page to appear functional while the backend erroneously accepts duplicate identities or allows a cross-account write. Conversely, a rejected malicious request should be documented as a successful security test when it matches the expected denial. Actual HTTP response codes, response bodies, and data-diff evidence should be preserved, with secrets redacted. The current inventory supports the design of these cases but contains no raw transcript of their execution.

The reset workflow requires particular caution because it is described as a demonstration without email authentication. A

suitable security regression would ensure that the demo path is disabled or replaced before production exposure. Recording a successful hash change under the demo policy must not be reported as secure ownership verification. Attempts should be conducted only on disposable accounts in an authorized local environment.

D. Learning and Progress-Persistence Tests

TABLE IX. LEARNING PROGRESSION, DURABILITY, AND BOUNDARY TEST DESIGN

Case	Setup and action	Expected behavior
LE-01	Attempt Try before Learn	Try remains locked
LE-02	Complete Learn twice	XP awarded once
LE-03	Submit 15-char reflection	Reject or keep incomplete
LE-04	Submit valid 16-char reflection	Allow Try completion if other conditions satisfied
LE-05	Attempt Build before Try	Build remains locked
LE-06	Complete all three phases	Topic receives 100 XP
LE-07	Refresh after saving draft	Local draft recovered in same profile
LE-08	Switch to another role	Old role progress not misattributed
LE-09	Clear localStorage	Describe and handle missing in-progress records

The test oracle for XP is deterministic, but the test oracle for learning achievement is not. LE-06 verifies an accounting rule, not skill mastery. Learning effects would require a separate assessment that does not reuse the application's same memorized questions. Streak validation additionally needs test dates around midnight, repeated actions within one day, a gap day, and timezone changes. Only code-level and browser-level tests can establish how the date parser actually behaves in these cases.

Progress-persistence tests should explicitly note which storage domain they exercise. A page refresh within the same browser is different from account login on another device; localStorage continuity does not entail cross-device sync. A successful course result on the backend should be queried again after a clean login. These observations would expose any mismatch between client phase flags and durable completion marks.

E. Assessment and Interview Reproduction

TABLE X. ASSESSMENT AND INTERVIEW TEST DESIGN

Case	Test input or scenario	Expected result / limitation
AS-01	Five known questions, three correct	Raw 60 and pass if $W \leq 2$
AS-02	Same answers, $W = 3$	Score 55 and fail
AS-03	Five correct, $W = 11$	Score 55 and fail under stated rule
AS-04	Repeat same question set with new shuffle	Correct answer text preserved
AS-05	Empty weak-skill history	Safe fallback, no divide-by-zero
AS-06	Direct Stage 3 route before Stage 2	Gate blocks access
IN-01	Paraphrase with synonyms not in dictionary	Record possible false negative
IN-02	Incorrect text containing all keywords	Record possible false positive
IN-03	Speech recognition unavailable	Manual text interaction remains available
IN-04	Nonterminating JS source	Isolation and timeout actually verified

AS-01 to AS-03 have derived expected numerical outcomes, but the individual application's runtime behavior remains unobserved. Test results should store an explicit distinction between calculated oracle and observed output. The complete question text and answer-key mapping should be recorded only in a controlled test artifact, not exposed to learners. Repeated option shuffles should be checked for correctness rather than evaluated with a spurious randomness 'accuracy' percentage.

For the interview, use a fixed career-stratified set of candidate answers written or rated by independent subject-matter experts. Include concise correct answers, verbose incorrect answers containing target terms, negations, partial credit cases, equivalent synonyms, empty strings, language variants, and speech-transcription errors. Run the exact same responses through the recorded normalization and alias dictionaries. A blinded rater should assign reference scores without seeing the system output, allowing disagreement to be measured rather than assumed away.

The coding runner also requires browser-specific testing. A UI indication of a timed-out program should be accompanied by evidence that the program ceased execution without freezing the main application, accessing restricted network resources, or retaining state for subsequent interview items. Browser sandbox policies are protective mechanisms, but their configured effect must be measured in the deployed code and relevant user agents.

F. Metrics and Statistics for Subsequent Evaluation

For a suite of independently executed cases, equation (14) defines the functional pass rate, where N_{executed} is the total number of genuinely run cases and N_{pass} is the number whose observed behavior matches the predefined oracle. Such a percentage cannot be called AI accuracy: it measures conformity of software behavior to chosen tests. The rate also depends strongly on which tests were selected. A test pass rate without a coverage description can be misleading.

For interview agreement, let s_i be the system's predicted mark and r_i an independently adjudicated human-reference mark for answer i . Mean absolute error is equation (15), and the proportion of system marks within tolerance τ points of reference is equation (16). Both require a disclosed answer corpus and a scoring rubric. A low MAE would indicate score agreement in that corpus, not automatically educational benefit. Confidence intervals and inter-rater reliability should accompany any serious evaluation.

For educational outcomes, a consenting participant's independent posttest-minus-pretest difference is equation (17). An interpretation of improvement requires a comparison condition and attention to pre-existing differences, practice effects, attrition, and the equivalence of test forms. Task completion rate is equation (18). Service responsiveness may be summarized using the 95th percentile in equation (19); functional coverage of specified paths can be tracked through equation (20), provided the feature universe is defined before testing.

$$\text{PassRate} = 100 \frac{N_{\text{pass}}}{N_{\text{executed}}} \quad (14)$$

$$MAE = \frac{1}{n} \sum_{i=1}^n |s_i - r_i| \quad (15)$$

$$Agreement_{\tau} = \frac{1}{n} \sum_{i=1}^n \mathbb{1} [|s_i - r_i| \leq \tau] \quad (16)$$

$$\Delta_i = Y_{i,post} - Y_{i,pre} \quad (17)$$

$$TaskSuccess = \frac{N_{completed}}{N_{attempted}} \quad (18)$$

$$Latency_{p95} = Q_{0.95}(t_{response} - t_{request}) \quad (19)$$

$$Coverage = \frac{|\mathcal{F}_{tested}|}{|\mathcal{F}_{specified}|} \quad (20)$$

No values are filled in for these proposed metrics because the source evidence contains neither participant observations nor timing samples. This is intentionally different from inserting a graph with imagined accuracy or performance values. The formulas establish exactly how future outcomes would be defined if measurements were later collected, while keeping completed and proposed work separate.

G. Documentation-Reported Regression-Test Outcome

The supplied inventory reports a single native Node test run in which 19 of 19 tests passed, with none failed or skipped. The stated categories comprise 4 backend integration, 5 assessment regression, and 10 learning utility tests. It also reports an elapsed time of about 14.08 seconds. There is no underlying command transcript, versioned commit, OS/hardware description, or standard-output log in the evaluated materials, so elapsed time should not be interpreted as a repeatable benchmark.

Figure 17 summarizes the provenance and limits of the available evidence. The implementation inventory reports system behavior and a 19-test pass summary; 13 screenshots were supplied, of which nine research-relevant views are retained in this manuscript. Neither source establishes that those tests were independently rerun here, or that learner outcomes or runtime performance were measured. The matrix uses categorical labels rather than unsupported numerical performance graphics.

EVIDENCE BOUNDARIES

Documented claims are not independent observations.

DOCUMENTED SYSTEM
8 career pathways; 64 topics reported
SUPPLIED INTERFACES
13 screenshots received; 9 reproduced here
AUTOMATED TESTS
19 of 19 passing tests reported, not rerun
EDUCATIONAL OUTCOMES
No participant study or measured gain

Evidence types—not performance accuracy or statistical results.

Fig. 17. Evidence-provenance matrix distinguishing documented implementation, supplied interface captures, reported automated tests, and outcomes not independently reproduced.

TABLE XI. EXISTING RESULT REPORT AND EVIDENCE LIMIT

Reported quantity	Value in supplied material	Independent status
Automated cases	19 executed; 19 passed	Not rerun
Backend integration	4 cases	Assertions unavailable
Assessment regression	5 cases	Assertions unavailable
Learning logic	10 cases	Assertions unavailable
Test elapsed time	Approximately 14.08 seconds	Environment unknown
Live browser completion	Not supplied	No observation
Learning-effect estimate	Not supplied	No participant study
Expert interview agreement	Not supplied	No reference corpus

H. Arithmetic Checks That Can Be Verified Without Running the App

Certain boundary properties are fully determined by the documented formulas. With $Q = 5$, raw marks are restricted to 0, 20, 40, 60, 80, and 100 before warning deductions. A user with three correct responses has the minimum raw score that reaches 60. Additional chargeable warnings lower the score in five-point increments. By construction, the final max operation prevents negative marks. As W grows without bound, final marks remain zero instead of becoming negative.

The fixed XP allocation similarly implies an upper bound of 100 XP per topic if each phase can complete once. With 64 documented topics, a purely arithmetic ceiling would be 6400 XP across all roles, but no user is shown to complete all roles, and the persistence semantics across careers are not independently established. This ceiling is not reported as an observed user score. A single role with eight topics has an arithmetic ceiling of 800 XP under the same once-only assumption.

The 32-bit FNV-1a mapping has at most 2^{32} outputs regardless of the potentially much larger number of possible input strings. The pigeonhole principle guarantees collisions across sufficiently many input combinations, although it says nothing about whether a collision has occurred in this project. The hash therefore cannot be used as a cryptographic authenticity proof. These analytic checks are valuable because they challenge overly strong language before empirical testing starts.

I. Recommended Controlled Usability Study—Not Yet Conducted

A later usability study should recruit a clearly defined population of novice learners, disclose inclusion criteria, and obtain appropriate consent. Participants could attempt registration, selecting a role, starting a roadmap topic, submitting an exercise reflection, finding weak-skill feedback, beginning an interview, and locating a certificate requirement. Observers should record whether each task is completed without assistance, the time taken, visible errors, help requests, and the user's comprehension of the result.

A comparison study could contrast the integrated workflow with a set of separate reference resources under a preregistered task protocol. Equalize the instructional content where possible and randomize task order or assignment to reduce confounding. Use a skills measure independent of the CareerAI question bank, and report attrition, uncertainty, and participant

characteristics only with proper consent. This design is a proposal, not a completed human-subject experiment; the current manuscript reports no invented sample size or percentage improvement.

Because 'job ready' is a consequential label, a separate study should ask role-specific expert judges whether the skills evidence displayed is sufficient for that statement. The default assumption should be that it is not validated until evidence exists. The impact of anti-cheat warnings on accessibility and honest learners also deserves a documented user study rather than relying on the presence of focus-change listeners.

IX. DISCUSSION OF DESIGN TRADEOFFS AND RESEARCH CLAIMS

A. What Integration Establishes

CareerAI combines multiple activities around a common role context. The reported interface provides a clear sequence of actions, and the technical inventory supplies enough algorithm detail to derive gate behavior. These properties are useful engineering outcomes: a developer can identify how a score leads to a new screen, where data are stored, and what must be reproduced to verify a result. Integration reduces the need to manually translate between unrelated websites, but whether learners perceive that advantage or complete more tasks remains an empirical question.

The architecture also makes a clear separation between formative practices and final course marks. Learning XP and drafts are convenient to store locally, while user records and marks are kept by a protected service. This split simplifies a prototype and makes local test scenarios straightforward, but it creates ambiguity in course completeness across devices. An integrated-looking UI can hide these state differences unless the design carefully explains them.

B. Consequences of Deterministic Personalization

Rule-based recommendation has advantages for auditability: given a response history and a stable question bank, one can reproduce weakness ratios and selection priority. No opaque model weights are required. This is valuable in a preliminary education system because users and reviewers can inspect why a stage was recommended. The price is limited expressiveness and sensitivity to sparse, poorly tagged, or repeatedly encountered items. Personalization must not be confused with semantic understanding.

The mock interview shares this tradeoff. Keyword dictionaries are fast, local, and interpretable but may misclassify nuanced responses. Browser speech recognition adds a convenient input method without guaranteeing equivalent accuracy across accents, background noise, or languages. Future machine-learning components would introduce model governance, dataset requirements, costs, and additional bias concerns; they should be justified by measured improvement over the transparent baseline rather than added for branding.

C. Data Minimization, Auditability, and Learner Autonomy

Assessment systems need enough data to explain a decision and permit correction. However, retaining full speech

transcripts, passwords, behavioral surveillance, or sensitive audit metadata without clear purpose increases privacy risk. The current database descriptions indicate completion records and activity history, while many learning details remain local. A future design could store minimized per-attempt evidence on the server, allow learners to view and appeal scores, and limit owner access to information needed for maintenance.

Anti-cheat features should be proportional to stakes. A self-guided practice stage with five short multiple-choice questions is not a secure licensure examination; forbidding clipboard operations and penalizing tab changes may disrupt accessibility while failing to prevent determined outside assistance. Clear warnings, an accessible practice mode, and human-review options may offer better educational value than treating every browser focus change as misconduct.

D. Certificate Trust and Employment Communication

A passport can provide a useful record of participation when its meaning is transparent. It should say what was completed, which rules were used, and what has not been certified. If the issuer wishes to make externally verifiable claims, it must bind them to server-approved evidence and cryptographic assertions. A local noncryptographic hash identifies an input; it is not a defense against forgery or modification. Clear semantics protect both learners and employers from overinterpreting the document.

Similarly, an outbound job-search link helps navigation but does not establish a relevant current opening or hiring eligibility. A learner should be told that vacancies are controlled by external sites and may be outdated, region-specific, or inaccessible without login. A future feature claiming direct job matching would require a separate, source-backed corpus and observable relevance metrics.

X. LIMITATIONS, THREATS TO VALIDITY AND ETHICAL ISSUES

A. Internal and Construct Validity

The largest internal-validity risk is that the implementation inventory is not a reproducible experiment. Described functions and a reported passing suite can be accurate while important browser paths remain broken. No user journey was rerun for this manuscript, and no raw software evidence was provided to confirm stated versions or exact dependencies. The word 'verified' is therefore reserved for arithmetic checks, official publication facts, and bibliographic facts independently confirmed from source records.

Construct validity is limited because the assessment scores are based on five items per stage and fixed thresholds. Without item analysis, content blueprint review, and reliability estimation, a score cannot be interpreted as stable occupational readiness. Concept matching compounds the problem by conflating mention of a term with understanding. Completion XP measures a software state transition, not learning. This paper distinguishes each measure's actual construct from the attractive labels appearing in the interface.



B. External, Ecological, and Statistical Validity

No participant sample, workplace criterion, job outcome, or industry benchmark was provided. The study therefore cannot generalize effectiveness to learners in different regions, skill levels, accessibility situations, or hardware environments. External job websites introduce additional variation beyond the application's control. No significance tests or confidence intervals are appropriate for missing outcome data, and the documentary test counts cannot meaningfully support statistical claims about learner ability.

Any later study using the same question bank for training and measurement risks contamination. Repeated exposure allows memorization, and weak-skill targeting can make later in-app scores rise without commensurate transfer to new problems. Independent evaluation items, blinded reference scoring, and retention tests are required to separate familiarization from genuine learning.

C. Security, Privacy, and Operational Threats

Potential failures include cross-account progress leakage on a shared browser, a forged course mark sent from the client, stale signed cookies remaining usable after logout, unvalidated reset behavior, leakage of admin activity logs, script execution escaping the intended sandbox, and inconsistent completion after backend errors. These are analysis targets, not confirmed exploitable vulnerabilities. The actual code and deployment must be inspected before any security finding is asserted as fact.

Personal data include account email, display name, usage history, and course marks. Audio dictation may involve browser or service-provider handling depending on the implementation and browser, and the record of third-party privacy impact is incomplete. User-facing documentation should state retention, consent, cross-device sync behavior, and appeal or correction processes. OWASP [12] and W3C [11] offer relevant control and data-model baselines without providing an automatic certification of this application.

D. Documentation Completeness and Authentic Figures

The eight technical diagrams are original reconstructions of the documented architecture and workflow, not screenshots of runtime execution. Figures 7–19, by contrast, are authentic user-supplied interface captures illustrating concrete states of the application. Because the source version, capture browser settings, and full interaction logs are unavailable for those images, they should not be treated as independently reproduced measurements or proof that every behind-the-scenes rule executes as documented.

The inventory reports an unimplemented AI/ML Engineer tile, a demo-only password reset, a link-based rather than API-backed job hub, and browser-local in-progress state. Explicit disclosure of these boundaries increases the scientific usefulness of the account. Overlooking them would make the manuscript more promotional but less reproducible.

XI. FUTURE DEVELOPMENT AND EMPIRICAL RESEARCH AGENDA

A. Source and Test Reproducibility

The first priority is to preserve a complete source release or authorized private evaluation snapshot at a named commit and include a lockfile, deterministic seed data, environment manifest, and captured test logs. Expand the reported 19-case suite with cross-account, course-mark, browser, code-runner, and persistence tests. Record each case's independent oracle, expected/observed output, pass/fail outcome, and runtime context, then report the resulting count without equating it to algorithmic accuracy.

A browser-driven integration suite should traverse the entire learner sequence using disposable identities. It should capture screenshots only after the screen is actually rendered, with timestamps and provenance. A second browser and a second device can test which progress values synchronize. Negative tests—failed stages, invalid forms, denied admin access, and backend outage—are as important as the happy path.

B. Educational and Assessment Validation

A content panel covering all eight career roles should review topic coverage, question correctness, skill tags, cognitive level, scenario authenticity, and alignment between the learning resources and the final gates. The five-question-per-stage design should be assessed for reliability, and later-stage weak-skill selection should be compared with nonadaptive or calibrated alternatives. A role-specific empirical instrument can measure transfer to unfamiliar problems after a reasonable interval.

Interview scoring should begin with a stable, annotated answer collection rated blind by at least two qualified reviewers. Measure inter-rater agreement before treating their consensus as a reference. Evaluate system–reference discrepancies with the metrics defined in Section VIII and report confusion patterns for concepts if the annotation scheme genuinely supports them. Only then consider a learned semantic evaluator, preferably retaining explainable rubrics and human appeal.

C. Credential Integrity and Cross-Device Progress

Progress persistence should move from modifiable local state to a server-backed, learner-owned data model with explicit versioning, concurrency behavior, and retention limits. Local drafts may still be cached for resilience, but passing results and eligibility should be verified by a protected service. An append-only attempt record would support longitudinal study and safe disputes. Data export and account deletion mechanisms should be considered as part of privacy by design.

To produce a genuinely verifiable passport, the issuer should bind the credential to trusted records, sign a canonical representation, expose a verification method, and support revocation or status inspection. The document should disclose its issuer, learning outcomes, assessment basis, and limitations. The W3C credential model [11] provides a standards-oriented path; adoption should be tested with independent verifiers rather than assumed from adding a QR code.

D. Human-Centered Use and Workforce Evidence

A user study should test whether the integrated path makes it easier to locate the next relevant learning activity, understand why an answer was marked wrong, and navigate the certificate requirements. It should include users with different prior skills, low bandwidth, mobile devices, and accessibility needs. Workload and comprehension may be more meaningful early outcomes than a prematurely advertised placement rate.

Employment effects require much stronger evidence: linked, consented outcomes over time, verified applications, comparable baseline groups, and careful treatment of selection bias. An outbound job-search click alone does not mean an application was submitted, an interview was obtained, or a job offer resulted. Future reporting should keep these events distinct and protect users from inflated claims.

XII. CONCLUSION

This manuscript examines the proposed mastery-gated, diagnostic-to-career framework through CareerAI, a documented React–Express–SQLite platform integrating eight role-based learning pathways, 64 reported curriculum topics, three practice phases, three five-item assessments, a structured interview, descriptive skills feedback, client-side PDF credentials, owner monitoring, and outbound vacancy search. The system's key personalization and scoring mechanisms are deterministic rather than trained AI models. Formal equations and boundary checks show how its gates and scores are expected to behave, while diagrams and workflow descriptions expose trust and persistence boundaries that a concise conference paper would leave underexplained.

The strongest numerical regression claim remains the project-inventory statement of nineteen passing automated tests; it was not independently rerun. The nine retained user-supplied screenshots improve visual traceability of the user journey, yet do not establish educational effectiveness, model accuracy, job-placement success, or cryptographic credential authenticity. The defensible contributions remain documented implementation mapping, comparison to relevant research, rule formalization, and a transparent evaluation specification.

REFERENCES

- [1] J. Hattie and H. Timperley, "The power of feedback," *Review of Educational Research*, vol. 77, no. 1, pp. 81–112, 2007, doi: 10.3102/003465430298487.
- [2] D. J. Nicol and D. Macfarlane-Dick, "Formative assessment and self-regulated learning: A model and seven principles of good feedback practice," *Studies in Higher Education*, vol. 31, no. 2, pp. 199–218, 2006, doi: 10.1080/03075070600572090.
- [3] B. J. Zimmerman, "Becoming a self-regulated learner: An overview," *Theory Into Practice*, vol. 41, no. 2, pp. 64–70, 2002, doi: 10.1207/s15430421tip4102_2.
- [4] V. J. Shute, "Focus on formative feedback," *Review of Educational Research*, vol. 78, no. 1, pp. 153–189, 2008, doi: 10.3102/0034654307313795.
- [5] R. R. Meijer and M. L. Nering, "Computerized adaptive testing: Overview and introduction," *Applied Psychological Measurement*, vol. 23, no. 3, pp. 187–194, 1999, doi: 10.1177/01466219922031310.
- [6] J. Dai, X. Gu, and J. Zhu, "Personalized recommendation in the adaptive learning system: The role of adaptive testing technology," *Journal of Educational Computing Research*, vol. 61, no. 3, pp. 523–545, 2023, doi: 10.1177/07356331221127303.
- [7] H. Abu-Rasheed, C. Weber, and M. Fathi, "Context based learning: A survey of contextual indicators for personalized and adaptive learning recommendations—A pedagogical and technical perspective," *Frontiers in Education*, vol. 8, 1210968, 2023, doi: 10.3389/feduc.2023.1210968.
- [8] N. Kamal, F. Sarker, A. Rahman, S. Hossain, and K. A. Mamun, "Recommender system in academic choices of higher education: A systematic review," *IEEE Access*, vol. 12, pp. 35475–35501, 2024, doi: 10.1109/ACCESS.2024.3368058.
- [9] F. Manganello, E. Rasca, A. Villa, A. Maddalena, and G. Boccuzzi, "Machine learning for post-diploma educational and career guidance: A scoping review in AI-driven decision support systems," *Frontiers in Education*, vol. 10, 1578979, 2025, doi: 10.3389/feduc.2025.1578979.
- [10] L. M. Oliva-Cordova, A. Garcia-Cabot, and H. R. Amado-Salvatierra, "Learning analytics to support teaching skills: A systematic literature review," *IEEE Access*, vol. 9, pp. 58351–58363, 2021, doi: 10.1109/ACCESS.2021.3070294.
- [11] World Wide Web Consortium, "Verifiable Credentials Data Model v2.0," W3C Recommendation, May 15, 2025. [Online]. Available: <https://www.w3.org/TR/2025/REC-vc-data-model-2.0-20250515/>.
- [12] OWASP Foundation, "Application Security Verification Standard, version 5.0.0," May 2025. [Online]. Available: <https://owasp.org/www-project-application-security-verification-standard/>.
- [13] F. Miao and W. Holmes, *Guidance for Generative AI in Education and Research*. Paris, France: UNESCO, 2023. [Online]. Available: <https://www.unesco.org/en/articles/guidance-generative-ai-education-and-research>.