IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, October 2026
Impact Factor: 8.2

AI-Driven Skill Assessment and Interview Preparation Platform Using A Multi-Agent Architecture

Nikhil D. Jagtap[1], Rahul J. Mane[2], Murlidhar C. Konda[3], Devanand B. Teli[4], Suraj G. Gaikwad[5]
[5]Professor, Department of Computer Science and Engineering
[1234]Students, Department of Computer Science and Engineering
Brahmdevdada Mane Institute of Technology

Abstract: *Students and early-career job seekers often prepare for recruitment through separate tools for resume review, mock interviews, coding practice, feedback, and career planning. This fragmented process makes it difficult to understand readiness across technical and communication skills or to convert assessment results into a coherent learning plan. This paper presents the design of an AI-driven Skill Assessment and Interview Preparation Platform that consolidates these activities in a modular web application. The proposed architecture combines a React-based user interface and Node.js/Express services with MongoDB persistence, Redis caching, and Firebase-based authentication. Specialized agents are defined for resume analysis, HR and technical interview practice, coding assessment, feedback, question generation, evaluation, and career-roadmap preparation. The workflow collects user inputs, analyzes resume content and assessment responses, presents structured feedback, and stores progress for subsequent sessions. The paper describes the problem, related work, system architecture, module responsibilities, data flow, implementation approach, and an evaluation plan. Since the supplied project report does not provide a reproducible dataset or measured benchmark results, no quantitative accuracy or improvement claims are made. The proposed evaluation uses functional test cases, response-time measurements, task-completion checks, and user feedback to assess the platform. Human review, privacy controls, and transparent feedback are emphasized because automated scores should support preparation rather than serve as definitive judgments of a candidate's employability.*

Keywords: Artificial Intelligence, Skill Assessment, Interview Preparation, Multi-Agent System, Resume Analysis, Coding Assessment, Personalized Learning, MERN Stack

I. INTRODUCTION

A. Background and Motivation

Recruitment commonly evaluates multiple dimensions: the relevance of a candidate's resume, subject knowledge, problem-solving ability, communication, and the capacity to explain decisions. Students may use one service to improve a resume, another to practice programming, and a third for mock interviews. Results from these tools are often disconnected, leaving learners to decide independently what to practice next. The project report underlying this work identifies this fragmentation as a central problem and proposes an integrated platform for preparation and skill assessment [1].

Recent advances in natural language processing and conversational systems make it practical to support text analysis, question generation, and structured feedback in software. However, automated assessment is imperfect: model outputs can be inconsistent, and scores may not correspond directly to human judgments or real hiring outcomes. Research on LLM-based resume matching has found that automated ratings should not be treated as interchangeable with human

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, October 2026
Impact Factor: 8.2

ratings [2]. Accordingly, the platform is positioned as a practice and learning aid, not as an autonomous hiring decision-maker.

B. Research Problem

The research problem is how to integrate resume analysis, interview simulation, coding practice, performance feedback, and career guidance into one manageable workflow while retaining modularity, traceability, and user control. The design must also handle practical concerns such as authentication, storage of personal information, service latency, and the limitations of AI-generated evaluations.

C. Contributions

- A modular, multi-agent system design for connected interview-preparation tasks.
- A unified workflow that links resume review, HR/technical interviews, coding assessments, feedback, and roadmaps.
- A web application architecture using React, Node.js/Express, MongoDB, Redis, and Firebase authentication as specified in the project report.
- An evaluation plan that separates functional verification from empirical claims about accuracy or candidate outcomes.

II. RELATED WORK

A. Resume Analysis and Skill Extraction

Resume analysis converts unstructured documents into fields that can be searched or compared. Common approaches include text extraction, normalization, keyword and phrase matching, named-entity recognition, and vector-based similarity. Li et al. investigated context-aware transformer models for resume classification and resume–job-description matching, showing the relevance of contextual representations for competency-related tasks [3]. Traditional TF-IDF representations remain useful as transparent baselines, but their lexical nature can miss synonyms and implied skills.

B. Conversational and Interview Systems

Transformer-based language models have become a foundation for many natural-language tasks [4]. BERT introduced bidirectional contextual representations [5], while Sentence-BERT provides sentence embeddings designed for efficient semantic comparison [6]. In an interview-practice setting, such methods may help organize role-specific questions and compare a response with expected concepts. They do not, by themselves, guarantee that a question is appropriate or that an evaluation is fair; rubrics and human review remain necessary.

C. Automated Coding Evaluation and Personalized Learning

Coding assessment can combine test-case execution with checks for correctness, edge cases, and computational efficiency. Execution results are more reproducible than free-form judgments, although test coverage limits what can be concluded about a solution. Personalized learning systems use observed performance to recommend subsequent learning activities. For this project, feedback and roadmap agents are designed to translate assessment observations into actionable next steps rather than claim to predict placement success.

D. Fairness, Explainability, and Research Gap

Automated systems can reproduce or amplify bias when they rely on incomplete data, proxy variables, or opaque scoring rules. Raghavan et al. examined claims and practices concerning bias mitigation in algorithmic hiring and highlighted the need for evidence and transparency [7]. Although the present system is intended for candidate self-preparation rather than employer selection, it may still influence confidence and learning choices. Its feedback should therefore identify the evidence behind recommendations, avoid unsupported personality judgments, and allow users to challenge or disregard suggestions.

The design gap addressed here is integration: resume feedback, interview simulation, coding evaluation, progress history, and career planning are represented as coordinated modules in a single application. This paper contributes an architecture and evaluation framework; it does not claim that integration alone proves superior learning outcomes.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, October 2026
Impact Factor: 8.2

III. PROBLEM STATEMENT AND OBJECTIVES

The project report identifies the following limitations in conventional preparation workflows:

- Preparation activities are spread across separate platforms, making progress difficult to consolidate.
- Practice interviews may not provide consistent, role-oriented feedback.
- Learners can struggle to identify gaps across resume quality, technical knowledge, coding, and communication.
- Generic recommendations may not translate assessment results into a practical learning sequence.
- Longitudinal progress tracking across different preparation activities is often limited. The main objectives of the proposed platform are to:
- Provide secure registration, login, and a personalized dashboard.
- Analyze uploaded resumes and identify relevant skills and possible gaps.
- Support HR and technical mock interviews through available interaction modes.
- Provide coding tasks and evaluate submitted solutions against defined checks.
- Generate structured feedback and role-oriented career roadmaps.
- Maintain assessment history so users can review progress over time.

IV. PROPOSED METHODOLOGY

A. End-to-End Workflow

The workflow begins when a user authenticates and selects a preparation activity. For resume analysis, the system accepts a supported resume file, extracts text, normalizes the content, and identifies skills and other relevant sections. For an interview session, the system selects or generates questions based on the chosen role and session type. User responses are recorded in the session context and evaluated against a rubric. In coding practice, a submission is checked using defined test cases and available execution results. The feedback module consolidates evidence from these activities, and the roadmap module suggests learning tasks associated with identified gaps. Results are stored so the dashboard can display history.

Fig. 1. Proposed system workflow: Authentication → Activity selection → Resume/interview/coding assessment → Evaluation → Feedback → Career roadmap → Progress dashboard.

B. Resume Analysis Pipeline

- Document intake: validate file type and size before processing.
- Text extraction: obtain machine-readable text; scanned documents may require OCR and should be flagged if extraction quality is poor.
- Preprocessing: normalize whitespace, remove repeated noise, and divide text into likely sections.
- Skill identification: detect terms and phrases using a maintained skill vocabulary and, where available, NLP-based entity extraction.
- Feedback generation: show detected skills and potential omissions as suggestions for user review.

A resume score should be treated as a heuristic indicator, not an objective measure of a person's ability. The interface should expose the factors used and avoid penalizing a user solely because a skill is described with different wording.

C. Interview and Question Generation

The Interview Agent supports HR and technical practice. Question selection may use the target role, user-selected topic, resume-derived skills, and previous session history. The evaluation module should use explicit dimensions such as relevance, technical correctness, clarity, and completeness. Where a language model is used, the prompt, rubric version, model version, and relevant inputs should be logged where appropriate for reproducibility and debugging. Voice or webcam functions should be used only with clear user consent and should not infer sensitive traits.

D. Coding Assessment

The coding module provides a programming task, accepts a submission, executes it in a controlled environment, and compares outputs with test cases. A useful result includes passed and failed tests, error information, and complexity guidance where that guidance can be justified. Code execution must be isolated from the application server and

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, October 2026
Impact Factor: 8.2

constrained by time, memory, filesystem, and network policies. If an external execution service is used, its availability and data handling should be documented.

E. Feedback and Roadmap Generation

The Feedback Agent consolidates evidence from resume, interview, and coding activities. The Report/Roadmap Agent then maps the observed gaps to concrete next actions, for example revising a resume bullet, reviewing a technical concept, or solving a problem of a particular difficulty. Recommendations should be traceable to the assessment evidence and should distinguish observed facts from generated suggestions.

V. SYSTEM ARCHITECTURE

The proposed design follows a layered architecture. The client layer is implemented in React and presents authentication, resume upload, interview sessions, coding activities, feedback, and dashboards. The application layer uses Node.js and Express to validate requests, enforce access rules, coordinate agents, and expose REST APIs. MongoDB stores user profiles, resume metadata, sessions, coding results, feedback, and roadmap records. Redis is intended for temporary state and frequently accessed data, such as session timers or rate-limit counters. Firebase Authentication provides the authentication mechanism described in the project report. Python-based AI services may be connected through service APIs where required.

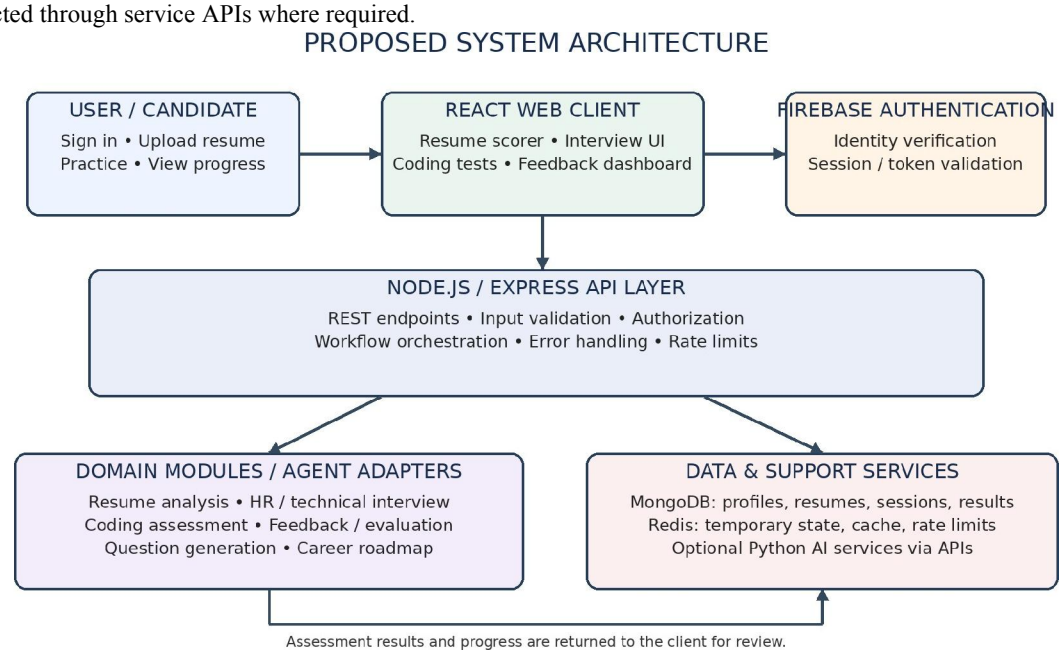


Fig. 2. Layered architecture: React client ↔ Express API and authorization ↔ domain modules/agent adapters ↔ MongoDB and Redis; Firebase Authentication supports identity verification.

A. Core Components

Component	Responsibility	Main Output
Authentication	Verify identity and protect user-specific resources	Authenticated session
Resume Agent	Extract text and identify skills and sections	Resume analysis
Interview Agent	Manage HR/technical practice sessions	Questions and session record
Coding Agent	Coordinate tasks, submissions, and test results	Coding assessment
Evaluation/Feedback	Apply rubrics and summarize evidence	Structured feedback
Roadmap Agent	Translate gaps into learning actions	Personalized roadmap
Dashboard	Present records and progress history	Progress summaries

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, October 2026
Impact Factor: 8.2

Admin/Operations	Monitor system behavior and manage permitted settings	Operational visibility

VI. TECHNOLOGIES USED

Technology	Purpose in the proposed system
React.js	Component-based user interface and dashboards
Node.js and Express.js	REST APIs, business logic, validation, and agent orchestration
MongoDB	Persistence for profiles, resumes, sessions, scores, and reports
Redis	Temporary state, caching, timers, and rate limiting where appropriate
Firebase Authentication	Identity and sign-in integration
Python / NLP / LLM services	Optional analysis, question generation, and feedback tasks
Docker	Repeatable packaging of services
AWS	Potential cloud deployment environment stated in the project report
Git and GitHub	Version control and collaborative development

The technology list reflects the submitted project report. The exact AI model, model provider, deployment configuration, and whether every planned service is active must be confirmed against the implementation before publication.

VII. IMPLEMENTATION DESIGN

A. Data Model

A practical data model separates identity from assessment records. Suggested collections include Users, Resumes, InterviewSessions, InterviewResponses, CodingTasks, CodingSubmissions, FeedbackReports, CareerRoadmaps, and ActivityEvents. Records should include timestamps and references to the owning user. Version identifiers for rubrics and AI configurations improve reproducibility. Sensitive files should not be embedded unnecessarily in general-purpose records; access-controlled object storage may be used if the deployed system requires it.

Collection	Representative fields
Users	userId, displayName, email, role, createdAt
Resumes	resumeId, userId, fileReference, extractedSkills, analysisVersion, createdAt
InterviewSessions	sessionId, userId, role, sessionType, questionSetVersion, status
CodingSubmissions	submissionId, userId, taskId, language, testSummary, submittedAt
FeedbackReports	reportId, userId, sourceSessionIds, rubricVersion, strengths, improvementAreas
CareerRoadmaps	roadmapId, userId, linkedFeedbackId, recommendedActions, createdAt

B. Representative API Design

Method	Endpoint (illustrative)	Purpose
POST	/api/resumes/analyze	Upload and analyze a resume
POST	/api/interviews/sessions	Create an interview practice session
POST	/api/interviews/sessions/{id}/responses	Submit an answer
POST	/api/coding/submissions	Submit code for evaluation
GET	/api/feedback/{id}	Retrieve a feedback report
GET	/api/roadmaps/current	Retrieve the user's current roadmap
GET	/api/dashboard/summary	Retrieve progress summary

These endpoints are proposed examples for describing the architecture; they should be replaced with the exact routes from the running application before final submission. Every endpoint must validate input, authenticate the caller, authorize access to the requested resource, and return clear error responses.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, October 2026
Impact Factor: 8.2

C. Security and Privacy

- Use server-side authorization checks; hiding a UI control is not sufficient access control.
- Limit file types and sizes, scan or safely process uploads, and avoid executing uploaded content.
- Protect secrets with environment variables or a secret manager and never commit credentials to source control.
- Apply least-privilege database access, rate limiting, secure transport, and retention rules for personal data.
- Obtain consent before recording voice or webcam input and provide a clear deletion or retention policy.
- Treat generated assessments as fallible and keep human review central to consequential decisions.

VIII. EVALUATION PLAN

The supplied black book describes multiple testing categories but does not provide a reproducible evaluation dataset or measured benchmark results. Therefore, the present paper defines an evaluation protocol rather than reporting unverified performance. Testing should be conducted on a documented version of the application using consented or synthetic test data.

Evaluation area	Method	Metric to report
Functional correctness	Run test cases for login, upload, interview, coding, feedback, and roadmap flows	Pass rate and failures by module
Resume extraction	Compare extracted fields with manually annotated test resumes	Precision, recall, F1, field-level error rate
Interview feedback	Have independent reviewers rate outputs using a fixed rubric	Agreement, rubric scores, error categories
Coding evaluation	Run known correct and incorrect submissions against tests	Test-case pass rate and false acceptance/rejection
Performance	Repeat representative API requests under recorded load	Median and p95 latency, throughput, error rate
Usability	Conduct task-based sessions with target users	Task completion, time, and user-reported usability
Privacy/security	Review access controls, file handling, logs, and retention behavior	Findings and remediation status

A. Reporting Rules

For each experiment, report the application version, hardware or cloud configuration, sample size, data source, inclusion criteria, measurement procedure, and limitations. Use separate training, tuning, and test sets if model fitting is performed. Avoid reporting a single aggregate score when errors differ materially across resume formats, roles, languages, or user groups. Where human ratings are used, document the rubric and report inter-rater agreement.

B. Results Status

No measured accuracy, response time, sample size, or improvement percentage is asserted in this draft because those values are not established in the supplied report. After tests are run, this section should be updated with a results table and a brief analysis of observed failures. Expected benefits must not be presented as experimental findings.

IX. DISCUSSION

The main design contribution is the integration of several preparation activities into a continuous feedback loop. A user can move from resume analysis to practice, review feedback, and receive recommended next steps without manually combining records from unrelated tools. Modular agents also allow individual functions to be tested and changed independently. This architecture is consistent with the project report's stated aim of supporting students, fresh graduates, and professionals preparing for career transitions [1].

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, October 2026
Impact Factor: 8.2

The system also has limitations. Resume extraction can fail on scanned or unusually formatted documents. Generated questions may be repetitive, ambiguous, or unsuitable for a selected role. Automated evaluation may reward wording rather than underlying knowledge, and code tests cannot prove correctness for all inputs. Recommendations can become stale if learning resources change. Cloud hosting, model inference, and video processing can increase operating costs. These limitations should be addressed through testing, transparent feedback, versioned rubrics, user controls, and periodic review.

The platform's impact on interview readiness remains an empirical question. A credible study could compare baseline and post-use assessment scores, use a control or comparison group where feasible, and measure user-reported confidence alongside performance. Any claim of improved placement outcomes would require longitudinal evidence and careful consideration of external factors.

X. FUTURE SCOPE

- Multilingual resume analysis and interview practice with evaluation adapted to language.
- More robust document processing, including OCR for scanned resumes and explicit extraction-confidence indicators.
- Adaptive question difficulty based on previous responses and a transparent skill taxonomy.
- Expanded coding support with sandboxed execution, additional languages, and test coverage reports.
- Progress analytics that compare repeated attempts while accounting for differences in task difficulty.
- Optional integrations with learning resources and job portals, subject to API terms and user consent.
- Accessibility improvements and mobile-friendly or native mobile interfaces.
- Independent evaluation of reliability, fairness, usability, privacy, and learning outcomes.

XI. CONCLUSION

This paper has presented the design of an AI-driven Skill Assessment and Interview Preparation Platform based on the AI Driven Skill Assessment System project report. The proposed system combines resume analysis, HR and technical interview practice, coding assessment, structured feedback, career-roadmap generation, and progress tracking in a modular web application. The architecture uses React, Node.js/Express, MongoDB, Redis, and Firebase authentication, with AI services integrated through well-defined module boundaries. Its intended value is to help learners understand skill gaps and organize preparation activities in one place. At the same time, automated scores and generated recommendations must be treated as fallible support. A publishable empirical version of this work requires implementation-specific evidence, documented test data, reproducible evaluation, and careful reporting of limitations. The next step is therefore to measure the system using the protocol described in this paper and replace proposed design details with verified implementation facts.

ACKNOWLEDGMENT

The authors acknowledge the Department of Computer Science and Engineering, Brahmdevdada Mane Institute of Technology, Solapur, and the faculty members who supported the academic project. The final acknowledgment should be checked and approved by the authors and project guide before submission.

REFERENCES

[1] N. D. Jagtap, R. J. Mane, M. C. Konda, and D. B. Teli, “AI Driven Skill Assessment System,” final-year project report, Department of Computer Science and Engineering, Brahmdevdada Mane Institute of Technology, Solapur, India, academic year 2024–2025. (Project report supplied by the authors.)

[2] S. Vaishampayan et al., “Human and LLM-Based Resume Matching: An Observational Study,” in Findings of the Association for Computational Linguistics: NAACL 2025, pp. 4823–4838, 2025, doi: 10.18653/v1/2025.findings-naacl.270.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, October 2026
Impact Factor: 8.2

[3] C. Li, E. Fisher, R. Thomas, S. Pittard, V. Hertzberg, and J. D. Choi, "Competence-Level Prediction and Resume & Job Description Matching Using Context-Aware Transformer Models," in Proceedings of EMNLP 2020, 2020. Available: https://aclanthology.org/2020.emnlp-main.679/
[4] A. Vaswani et al., "Attention Is All You Need," in Advances in Neural Information Processing Systems, vol. 30, 2017.
[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in Proceedings of NAACL-HLT, pp. 4171–4186, 2019.
[6] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks," in Proceedings of EMNLP-IJCNLP, pp. 3982–3992, 2019.
[7] M. Raghavan, S. Barocas, J. Kleinberg, and K. Levy, "Mitigating Bias in Algorithmic Hiring: Evaluating Claims and Practices," in Proceedings of the ACM Conference on Fairness, Accountability, and Transparency, pp. 469–481, 2020.
[8] Meta, "React Documentation." [Online]. Available: https://react.dev/
[9] OpenJS Foundation, "Node.js Documentation." [Online]. Available: https://nodejs.org/docs/
[10] MongoDB, "MongoDB Manual." [Online]. Available: https://www.mongodb.com/docs/manual/
[11] Redis, "Redis Documentation." [Online]. Available: https://redis.io/docs/latest/
[12] Google, "Firebase Authentication Documentation." [Online]. Available: https://firebase.google.com/docs/auth
[13] Docker, "Docker Documentation." [Online]. Available: https://docs.docker.com/
[14] Amazon Web Services, "AWS Documentation." [Online]. Available: https://docs.aws.amazon.com/