Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

AI-Based Mock Interview, Aptitude and Placement Prediction Platform for Engineering Students

Jayanth s swamy[1], Darshan MP[1], Samarjit Nayak A, Tejas N L [Guide / Co-author Ms. Madhushree][1]
[1]Department of [Computer Science and Engineering], [G Madegowda Institute of Technology], [Bharathinagar, Mandya dist, India]

Abstract - Campus placement outcomes for engineering graduates depend on aptitude, technical depth, communication and academic standing, yet most students prepare without structured, personalised feedback. This paper presents an integrated web platform that combines (i) a rule-based adaptive aptitude engine that adjusts question difficulty from recent accuracy and response time, (ii) an AI-assisted mock interview module that accepts typed and live voice responses and scores them across multiple competency dimensions, (iii) a machine-learning placement-readiness predictor, and (iv) a personalised 7/14/30-day preparation planner. The system is implemented with React, FastAPI, SQLAlchemy and scikit-learn, together with a separate Express service that sends voice-interview transcripts to a large language model and falls back to a heuristic scorer when the model is unavailable. On a 1,000-record synthetic dataset, a logistic-regression model reached 94.5% accuracy, a 96.2% F1-score and 0.985 ROC-AUC on a held-out 20% split, outperforming a random-forest baseline (91.5% accuracy). Because the labels are synthetic, these figures validate the machine-learning pipeline rather than real-world predictive power; validation on institutional placement records is identified as the principal next step.

Keywords - placement readiness; mock interview; adaptive testing; large language model; logistic regression; FastAPI; React

I. INTRODUCTION

Campus recruitment is the dominant route to first employment for engineering graduates in India, and recruiters typically screen candidates through an aptitude test, one or more technical interviews and an HR round. Students are expected to prepare for all three, but the preparation resources available to them are fragmented: question banks that do not adapt to the learner, mock interviews that depend on the availability of seniors or faculty, and no single indicator of overall readiness. As a result, weak areas often become visible only after a failed selection round.

Recent progress in large language models (LLMs) [9], [10] and browser-based voice agents makes it practical to offer interview practice on demand, while mature open-source web and machine-learning frameworks [2] lower the cost of building an end-to-end service. This work describes an integrated platform that brings these elements together for a single learner journey: profile creation, adaptive aptitude practice, text and voice mock interviews, readiness estimation and a personalised study plan.

The main contributions of this paper are:

- a rule-based adaptive aptitude engine that raises or lowers difficulty from a sliding window of recent answers and response time, and that reports topic-level strengths and weaknesses;
- a seven-dimension weighted rubric for typed interview answers and a five-category LLM rubric for live voice interviews, with score validation for typed answers and graceful fallback to a local heuristic scorer when the LLM is unavailable;
- a placement-readiness pipeline with an interpretable rule-based score for the dashboard and a trained logistic-regression predictor compared against a random-forest baseline; and
- a modular architecture (React front end, FastAPI core service, Express scoring service) with role-based access for students and administrators.

II. RELATED WORK

Adaptive testing. Item response theory and computerised adaptive testing select items whose difficulty matches the examinee's estimated ability, reducing test length while maintaining measurement precision [4], [5]. Full IRT calibration needs large response datasets; practical learning platforms therefore often use simpler rules that promote or demote difficulty from recent performance. The engine in this work follows the latter, lightweight approach.

Automated interview and answer evaluation. LLMs prompted with an explicit rubric have been shown to agree with human preferences at a level comparable to agreement between human raters, which has made "LLM-as-a-judge" a common technique for scoring free-text responses [6]. Reliable use requires constrained output formats and validation of the returned scores, both of which are incorporated in the present design.

Placement prediction. Studies on campus-placement prediction commonly model academic and skill attributes with classical classifiers such as logistic regression [3] and ensemble methods such as random forests [1], using libraries such as scikit-learn [2]. The present work follows this convention, but additionally couples the predictor to live signals from aptitude and interview practice and to a remediation planner.

III. SYSTEM ARCHITECTURE

The platform follows a three-tier design (Fig. 1). The presentation tier is a single-page application built with React 18, Vite, Tailwind CSS, Recharts and Framer Motion. The core service is a FastAPI application exposing REST endpoints under /api/v1 [8], with SQLAlchemy models persisted in

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

SQLite (switchable to PostgreSQL through the connection string). Authentication uses signed JSON Web Tokens [7] and distinguishes student and administrator roles. A separate Node.js/Express service handles live voice interviews: it stores session transcripts, calls the LLM for scoring and returns a structured report. Voice capture and speech synthesis for the live interview are delegated to the Vapi web SDK.

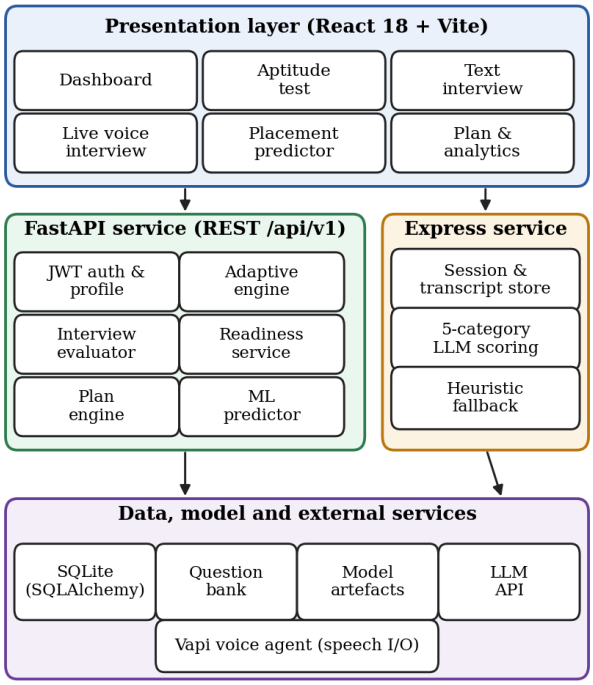


Fig. 1. Layered architecture of the proposed platform.

Table I summarises the technology stack. The model artefacts produced by the training pipeline (the classifier and the fitted preprocessing object) are serialised with joblib and loaded by the prediction endpoint at run time.

TABLE I. TECHNOLOGY STACK

Layer	Technologies
Front end	React 18, Vite 5, Tailwind CSS 4, React Router 6, Axios, Recharts, Framer Motion
Core API	Python, FastAPI, Pydantic v2, SQLAlchemy 2, JWT (python-jose), passlib
ML	scikit-learn, pandas, NumPy, joblib
Scoring service	Node.js, Express; Gemini 2.5 Flash or an OpenAI-compatible gateway
Voice	Vapi Web SDK (real-time speech I/O)

IV. METHODOLOGY

A. Student profile and data model

Each student maintains a profile containing CGPA, school-level percentages, active backlogs, target company tier, and collections of skills (with proficiency), projects, internships and certifications. A résumé in PDF or DOCX format can be uploaded. Aptitude attempts, answers, interview sessions, responses and generated plans are stored against the user identifier and drive the dashboard and analytics views.

B. Adaptive aptitude engine

The question bank covers four categories: quantitative aptitude, logical reasoning, verbal ability and data interpretation, each tagged by topic and by one of three difficulty levels (Easy, Medium, Hard). A session starts at Medium. After each answer the engine inspects the last three responses and applies the rules in Table II, which avoids oscillation caused by single outcomes.

TABLE II. DIFFICULTY TRANSITION RULES

Action	Condition
Promote	Last two answers correct; or window accuracy ≥ 75% and the last answer took < 40% of the time limit
Demote	Last two answers incorrect; or three answers in the window with accuracy ≤ 33%
Hold	Otherwise (a single result never changes level on its own)

The next question is chosen from the unasked pool by (1) restricting to the target topic when one is specified, (2) matching the current difficulty exactly, and (3) otherwise falling back to the nearest available difficulty. After a session, topic accuracy is aggregated: topics at or above 70% are labelled strengths, topics below 60% are weaknesses, and the three weakest topics generate text recommendations (below 50%: revise fundamentals; 50–70%: timed practice).

C. Typed AI mock interview

Questions are generated for an interview type (technical or HR), target role and difficulty through a provider abstraction that supports a mock provider, an OpenAI-compatible provider and a Gemini provider, so that the system remains usable without external keys. Each answer is scored on seven dimensions on a 0–10 scale and combined as

$$S = \Sigma\, w_k \cdot s_k\,, \quad k = 1 \ldots 7 \qquad (1)$$

with the weights in Table III. Returned scores are clamped to the valid range, missing dimensions receive neutral defaults, and absent strengths, weaknesses or improvement points are replaced with generic guidance, so that malformed model output cannot break the report. When no LLM is configured, a local heuristic evaluator derives the dimension scores from simple answer features: length, presence of domain terms, sentence punctuation, hesitation phrases (e.g., “not sure”) and structure markers (e.g., “first”, “for example”, “because”).

TABLE III. ANSWER-EVALUATION DIMENSIONS AND WEIGHTS

Dimension	Weight
Technical correctness	0.25
Relevance	0.20
Completeness	0.15
Answer structure	0.15
Clarity	0.10
Communication	0.10
Confidence indicators	0.05

D. Live voice interview and LLM scoring

In the live mode, the browser connects to a Vapi voice agent, shows an amplitude-driven waveform and a streaming

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

transcript, and supports microphone mute. When the call ends, the transcript is posted to the Express service, which prompts the LLM (temperature 0.2) to return strict JSON containing an overall score, five category scores (communication, technical knowledge, problem solving / STAR articulation, confidence and clarity, domain depth), three strengths and three improvement areas. The service tries the primary OpenAI-compatible gateway, then Gemini 2.5 Flash, and finally a deterministic heuristic scorer based on the amount of candidate speech. The dashboard readiness value is then updated by a weighted blend of the previous value and the new interview score:

$$R' = clip(\ round(0.35 \cdot R + 0.65 \cdot V),\ 50,\ 98\ ) \qquad (2)$$

where R is the previous readiness score and V the overall voice-interview score.

E. Rule-based readiness score

For the dashboard, a transparent additive score is computed from the student's record:

$$R = clip(\ round(\ 3C + 0.25A + 0.25I + P - 5b\ ),\ 10,\ 99\ ) \qquad (3)$$

where C is CGPA (0–10), A the mean aptitude percentage, I the mean completed-interview score (0–100), b the number of active backlogs, and P (0–20) rewards profile completeness (10 points for an uploaded résumé; 10 points for at least three skills, 5 otherwise). Scores of 80 or more, 60–79 and below 60 are labelled high, moderate and developing readiness. Factor-level notes (positive or negative) and targeted recommendations are generated from the same inputs.

F. ML placement predictor

Dataset. No institutional placement records were available, so a synthetic dataset of 1,000 students (708 labelled placed) was generated with a fixed seed. Eleven attributes are drawn from uniform or categorical distributions: CGPA, aptitude, technical, interview, communication and coding scores, counts of projects, internships and certifications, attendance and backlogs. The placement label is produced from a weighted performance score with Gaussian noise $\varepsilon \sim N(0, 4^2)$:

$$Z = 0.25(10C) + 0.20T + 0.20K + 0.15A + 0.10I + 0.05M + 3p + 5n - 12b + \varepsilon \qquad (4)$$

where T, K, A, I and M are technical, coding, aptitude, interview and communication scores, and p, n, b are project, internship and backlog counts. A student is labelled placed if $Z \geq 68$. The data file is explicitly marked as synthetic.

Pre-processing. Three engineered features are added: an academic–technical interaction (C/10 × T/100), the mean of the five skill scores, and an experience score (2p + 3n + 1.5 × certifications). Features are median-imputed and standardised; the data are split 80/20 with stratification (random state 42).

Models. A logistic-regression baseline [3] and a random forest with 100 trees and maximum depth 8 [1] are trained; the model with the higher F1-score on the test split is saved. For a student, the predicted probability is reported as a readiness percentage together with factor impacts.

G. Preparation planner, analytics and administration

The planner builds a 7-, 14- or 30-day schedule of two tasks per day from a library of prioritised activity blueprints, informed by weak aptitude topics, interview feedback and predictor output. Analytics pages chart score progression, accuracy trends and category mastery. An administrator portal summarises batch performance, the student registry, attempt statistics and question-bank management, and is protected by role-based authorisation.

V. RESULTS

A. Predictor performance

Table IV and Fig. 2 compare the two classifiers on the 200-student test split (142 placed, 58 not placed). Logistic regression achieved the best F1-score and was selected by the training script. Five-fold stratified cross-validation on all 1,000 records gave a mean F1 of 0.943 ± 0.014 for logistic regression and 0.921 ± 0.011 for the random forest, consistent with the single-split ranking.

TABLE IV. TEST-SET PERFORMANCE (n = 200)

Metric	Logistic Reg.	Random Forest	Δ
Accuracy	94.50	91.50	+3.00
Precision	95.17	90.32	+4.85
Recall	97.18	98.59	−1.41
F1-score	96.17	94.28	+1.89
ROC-AUC	98.51	96.69	+1.82
5-fold CV F1	94.3	92.1	+2.2

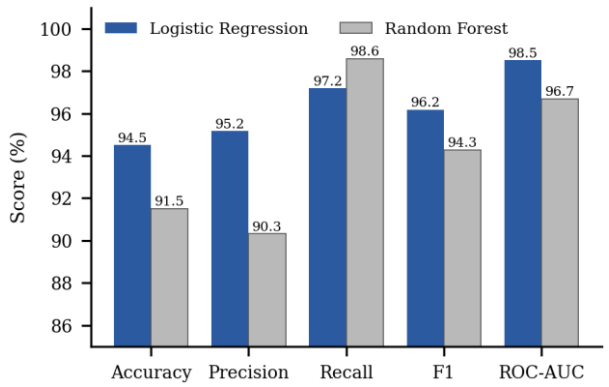


Fig. 2. Test-set metrics of the two classifiers (%).

The confusion matrices (rows: actual not placed / placed; columns: predicted) are [[51, 7], [4, 138]] for logistic regression and [[43, 15], [2, 140]] for the random forest. The random forest's higher recall comes at the cost of more false positives (15 versus 7), i.e., it more often labels an unprepared student as ready, which is the less desirable error for a readiness tool.

B. Factor influence

The largest standardised logistic-regression coefficients were backlogs (−3.10), experience score (+1.28), internship count (+1.26), overall skill average (+1.15), CGPA (+1.08), coding score (+1.04) and project count (+0.99). The sign and ranking agree with the label-generation formula, with backlogs the dominant negative factor. Certification count and attendance do not enter the label formula directly; their small

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

coefficients (−0.38 and +0.20) reflect collinearity with the engineered experience score and sampling noise, and should not be interpreted causally.

C. Functional verification

The repository includes automated tests for authentication, profile CRUD, aptitude scoring, the adaptive flow, interview evaluation, the ML predictor, plan generation, analytics and administrator authorisation (the project documentation reports all 51 tests passing). In our re-run, the seven unit tests of the adaptive engine passed, and re-training the predictor reproduced the metrics in Table IV exactly.

VI. DISCUSSION AND LIMITATIONS

Synthetic labels. Because the placement label is a near-linear function of the same features, a linear model is expected to fit it well, and the margin over the random forest is a property of the data generator rather than evidence that logistic regression is better for real placement data. The reported accuracy therefore demonstrates a working, reproducible pipeline; it should not be read as the platform's ability to predict actual hiring outcomes.

Heuristic scoring. The local interview evaluator relies on length and keyword cues, which are easy to game and do not verify technical correctness. The Express fallback scorer likewise yields scores bounded to a favourable range from the amount of speech alone. These paths exist for availability; LLM-based scoring should be preferred, and its agreement with human interviewers has not yet been measured.

Readiness defaults. When a student has no aptitude or interview history, the rule-based score currently substitutes placeholder baselines, which can overstate readiness for new users. Showing an "insufficient data" state instead would be more honest.

Persistence and scale. The voice-scoring service keeps sessions in memory, so they are lost on restart, and the core service uses SQLite. No usability study or classroom trial has yet been conducted.

VII. CONCLUSION AND FUTURE SCOPE

This paper presented an end-to-end placement preparation platform that unifies adaptive aptitude practice, typed and voice mock interviews with LLM-assisted scoring, an interpretable readiness score, a machine-learning predictor and a personalised study planner. A logistic-regression predictor trained on synthetic data reached 94.5% accuracy and 0.985 ROC-AUC, and the surrounding services were verified through automated tests.

Future work includes: (i) training and calibrating the predictor on anonymised institutional placement records with temporal validation; (ii) measuring agreement between LLM scores and faculty interviewers and adding a human-review loop; (iii) replacing rule-based difficulty adjustment with IRT-calibrated item parameters once enough responses are collected; (iv) persistent storage for voice sessions and a PostgreSQL deployment with migrations; (v) acoustic features such as pace and pause analysis; and (vi) an in-browser coding assessment. A controlled study with students, measuring score improvement and placement outcomes, would establish the platform's educational value.

REFERENCES

[1] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[2] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[3] D. W. Hosmer, S. Lemeshow and R. X. Sturdivant, *Applied Logistic Regression*, 3rd ed. Hoboken, NJ, USA: Wiley, 2013.

[4] F. M. Lord, *Applications of Item Response Theory to Practical Testing Problems*. Hillsdale, NJ, USA: Lawrence Erlbaum, 1980.

[5] H. Wainer *et al.*, *Computerized Adaptive Testing: A Primer*, 2nd ed. Mahwah, NJ, USA: Lawrence Erlbaum, 2000.

[6] L. Zheng *et al.*, "Judging LLM-as-a-judge with MT-Bench and Chatbot Arena," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2023.

[7] M. Jones, J. Bradley and N. Sakimura, "JSON Web Token (JWT)," IETF RFC 7519, May 2015.

[8] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, Univ. California, Irvine, CA, USA, 2000.

[9] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[10] Gemini Team, Google, "Gemini: A family of highly capable multimodal models," arXiv:2312.11805, 2023.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)