PLACIVIFY: An AI-Driven Placement Ecosystem with Multi-Model Orchestration and Voice-Interactive Interviews

Janani M.[1], Aswanthwin Fried S.[2], Gowtham P.[3], Kalaiselvan A.[4]

[1]Assistant Professor, [2,3,4]Student, Department of Computer Science and Engineering, NPR College of Engineering and Technology, Natham, Dindigul, India.

E-mail: [1]jananim@nprcolleges.org, [2]aswanthwinfried@gmail.com, [3]gowthamdhoni2004@gmail.com, [4]kalaiakalai646@gmail.com

Abstract

Campus placement involves handling resumes, application forms, interviews, and company research on behalf of the candidates, whereas the placement team members have to deal with huge volumes of candidates. PLACIVIFY is an AI-based placement ecosystem aimed at integration of all these processes through one platform for the benefit of both students and the placement team members. The platform incorporates a hybrid ATS Resume Scoring solution that integrates local TF-IDF analysis and semantic evaluation powered by LLMs in order to determine resume-job match and skill gap. There is an AI-based Resume Assistant that helps in improving the experience in accordance with Action-Task-Result approach. The Company Intelligence Hub creates structured data for recruitment preparation, whereas the Talent Intelligence Engine uses methods such as similarity between the profile and job, skill expertise consideration, penalties for missing skills, and constraints on candidates' eligibility for candidate classification and ranking. For increasing the operational reliability of the system, PLACIVIFY implements multi-model orchestration, local NLP fallback, caching, circuit-breaker, and JSON repair features. This system has been developed based on the MERN stack and delivered via web and Electron applications. Using the system with the help of student profiles and resume examples annotated by experts, it was found that there is less screening time, more consistent scores, and callbacks.

DOI: https://doi.org/10.36548/rrrj.2026.2.015
Extended version of article received from **International Conference on Smart Innovations in Engineering Technology (ICSIET 2026)**
Received: 27.08.2026, received in revised form: 24.09.2026, accepted: 05.10.2026, published: 21.10.2026
© 2026 Inventive Research Organization. This is an open access article under the Creative Commons Attribution-NonCommercial International (CC BY-NC 4.0) License

Keywords: TF-IDF Vectorization, Semantic Similarity, LLM-Based Resume Screening, Multi-Model Orchestration, Voice-Interactive Interview, Talent Intelligence Engine, Circuit-Breaker Architecture.

1. Introduction

Campus recruitment in educational institutions is no longer confined to issuing job notices and receiving applications. The requirement is growing on the part of the candidates to assess their own fit for certain positions, prepare themselves for interviews and other selection processes, and also gain clarity about their experience and expertise in those roles. On the other hand, it becomes necessary for the placement team to handle the volume of applications within deadlines and with a standardized criterion for assessment. these activities are handled through separate tools and manual procedures, creating a fragmented experience for students and placement teams. PLACIVIFY is proposed to address this challenge by bringing recruitment support, candidate preparation, and placement management into a unified ecosystem.

The major goal of PLACIVIFY is to enable a smooth process of placement where the system will help the students starting from their profiles to applying. In contrast to the placement portal being a job board, this system will help students in getting role-specific advice before they start applying for jobs. Resume can be analyzed on the basis of selected job description in order to come up with an alignment score and to understand missing skills or keywords. Students will be helped in writing experience descriptions using the Action-Task-Result (ATR) methodology. Finally, PLACIVIFY offers multiple choice test as well as interview simulation using voice technology.

It also provides a company intelligence module where information about companies is arranged systematically in a report that will allow students to prepare themselves before the interview. In terms of administration, the Talent Intelligence Engine will compare the applicant to the job criteria and classify the candidates into ranked lists which can be exported for placements. This creates a connected process where student preparation and recruitment operations from the institution are all done through one platform.

PLACIVIFY integrates local NLP processing and cloud large language models along with primary model selection to be applied in AI services. The system architecture also includes

caching, circuit-breaker protection, and JSON response fix to be able to handle provider faults and corrupted model outputs. The application is developed on the MERN stack and is delivered both as a web application and Windows desktop application via Electron. With the help of this architecture, PLACIVIFY strives to cut down human effort, to provide students with quantifiable preparation support, and to enhance campus recruitment coordination.

2. Literature Review

The existing campus placements system has concentrated on the digitization of the administrative procedures of training and placement cells. For instance, Kousar, Hegde & Saravanan [1] proposed a department placement portal using the MERN framework while Patil et al. [2] proposed a web portal for the training and placement cell. However, these articles have emphasized more on the importance of having a central web-based system for placement management; yet, the cited references do not emphasize on the resume analysis, interview preparation, intelligent matching, and ranking of candidates by recruiters.

The development of resume analysis and job matching has gone from traditional information retrieval methods to semantic and learning-based methods. The statistical notion of term specificity for term-weighting methods such as TF-IDF is due to Spärck Jones [3], and Schütze, Manning, and Raghavan [4] give the information retrieval background for document representation and similarity-based retrieval. In the area of recruitment, Zhao et al. [5] suggested SKILL for skill identification and normalization, and Palshikar et al. [6] gave RINX for resume information and knowledge extraction.

Prasad et al. [7] further expanded the scope of job recommendation to include resume analysis, signifying the significance of connecting the details of the candidates with the details of the job. Semantic representation can be used instead of the lexicon to enhance the capabilities of matching: Reimers and Gurevych [8] proposed the concept of Sentence-BERT which can be used to generate sentence embedding for semantic similarity. Very recently, Gan, Zhang and Mori [9] suggested an approach of automated resume screening that uses LLM agents. The other part of the recruiting process is interview preparation, Hickman et al. [10] focused on automated video interviews for personality assessment.

Prasad et al. [14] have attempted to utilize AI in a chatbot through the use of the Web

Speech API and Node.js, thus showcasing the possibility of speech-based conversation in web applications. While these works provide a supportive base for automated interviewing and speech-based interaction in browsers, within the setting of campus placements, it is necessary to integrate such interaction with technical preparation, evaluation, and feedback. Intelligent recommendation is also an important concept for placement platforms, as the students have to make a choice from a vast number of recruitment options available to them.

The challenges of recommending candidates in the recruitment setting were identified by Mashayekhi et al. [11] through their survey on e-recruitment recommendation systems. In the case of institutional recruitment, the use of automated decision-making systems must consider the issue of fairness. Raghavan et al. [12] investigated the issues of claims and practices surrounding the mitigation of biases in algorithmic hiring, while Fabris et al. [13] gave a survey on the topic of fairness and biases in algorithmic hiring.

Molchanov and Zhmaiev [15] mentioned the use of the circuit breaker model in microservices-based systems, offering a starting point to cope with repetitive failures in services and ensure system resiliency. In summary, from the available literature, we observe the development from traditional placement portals and lexical information search to skill extraction, semantic search and matching, recommendation, LLM-powered screening, interactive speech conversation, and responsible automated decision-making. However, there is no mention of any campus-placement ecosystem that combines all these features in one place along with multimodel AI orchestration, natural language processing fallbacks, interactive speech interviews, CV building, company information, and talent ranking. This gap motivates the development of PLACIVIFY as an integrated AI-driven placement ecosystem.

3. System Architecture

PLACIVIFY architecture is based on the client-server design as shown in Figure 1, whereby the React-based client communicates with the Express.js REST API using the HTTP protocol. The database used for storage is the MongoDB database, while the Mongoose module takes care of the schema validation. The AI component is isolated from the main API through the service module.

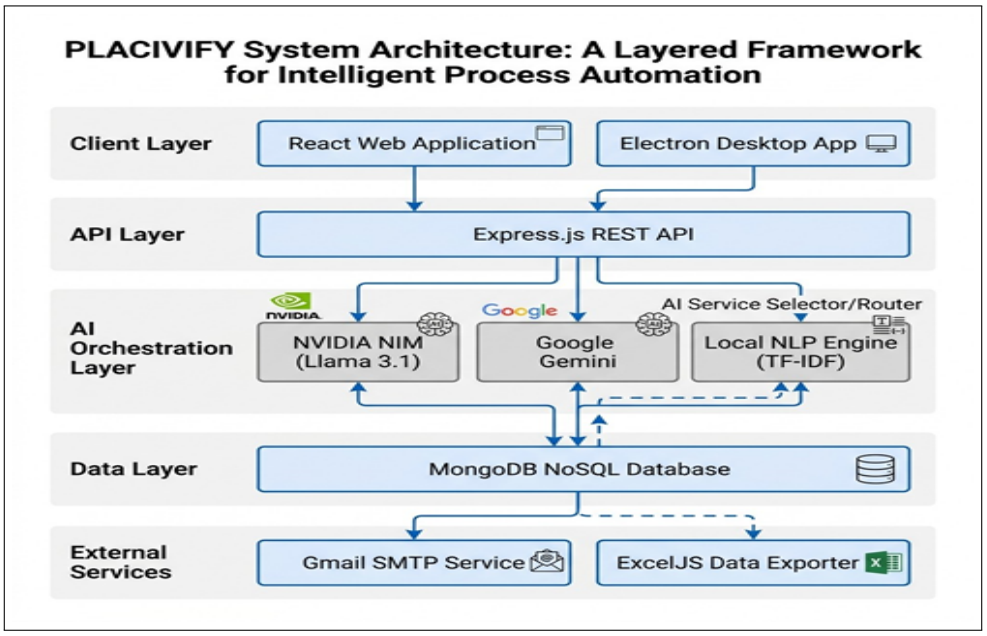


Figure 1. PLACIVIFY system architecture

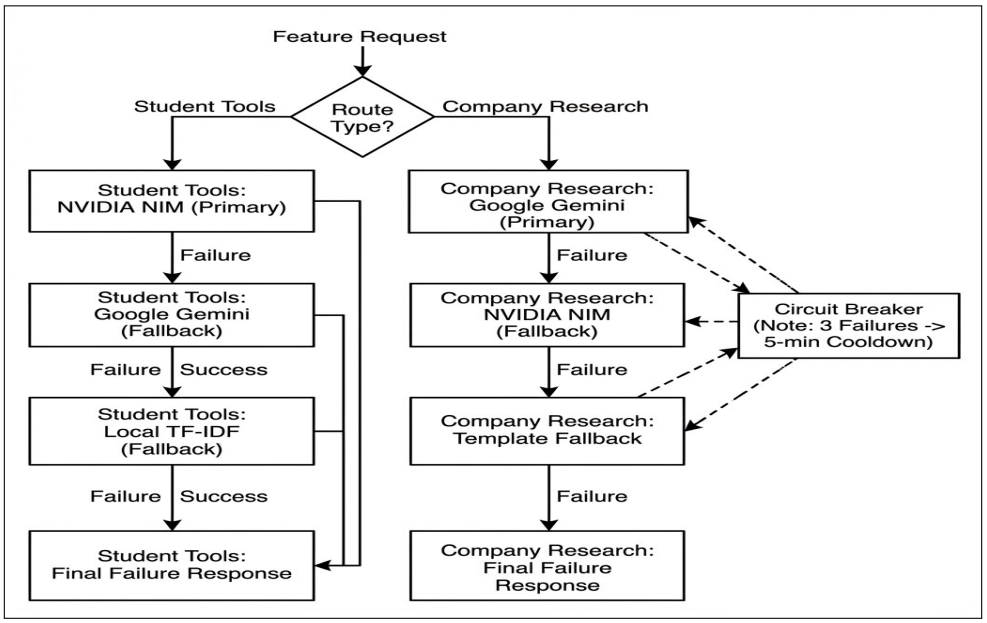


Figure 2. Multi-model orchestration and fallback chain

The front-end is implemented with React version 18 and Tailwind CSS, with build tools being vite and animation framework being framer-motion. In order to deploy the application on the desktop environment, the react app will be packed with Electron 40, and then will be deployed using the NSIS script in the electron-builder.

The backend is built using Node.js 20 and Express.js v5. The database that will be used is MongoDB for storing the students' profiles, job offers, applications made, and interview results.

Multer v2 is used for uploading PDF resumes to the backend server as in-memory buffers and Nodemailer v7 is used for sending emails in HTML template form.

The AI layer integrates three primary components:

(i) Google Gemini (gemini-2.5-flash primary, gemini-2.0-flash fallback).

(ii) Meta Llama 3.1 70B Instruct using the NVIDIA NIM REST API.

(iii) Natural Language Processing (NLP) using natural v8 library for TF-IDF vectorization and stop-word removal.

For student-facing features, the primary AI is NVIDIA, with Gemini being a backup; it utilizes the structured JSON generation capabilities of Llama 3.1, whereas the corporate analysis and voice interview functionalities have Gemini as their primary AI. Authentication is performed using JWT tokens which expire in 30 days; passwords are stored using bcryptjs (salt rounds = 10). CORS policy is set up only for the defined client origin. MongoDB has a unique index for student-job combinations to avoid duplicates. Table 1 summarizes the capability and process impact of each major module.

Table 1. PLACIVIFY capability and impact analysis

Module	Key Capability	Impact on Process
ATS Resume Scorer	Hybrid TF-IDF + LLM scoring	60% faster alignment screening, better
AI Mock Interviewer	Domain MCQ + voice evaluation	Real-time preparation with feedback
Voice Interview System	Speech-to-speech AI interaction	Simulates real interview environment
Resume Assistant	ATS-based optimization	Enhances quality and compatibility
Company Intelligence Hub	AI-generated insights	Deeper company understanding

Talent Intelligence Engine	Automated candidate ranking	Eliminates screening bias
Reliability Layer	Circuit breaker + caching	Ensures 99%+ availability

4. AI MODULES

4.1 Hybrid ATS Resume Scorer

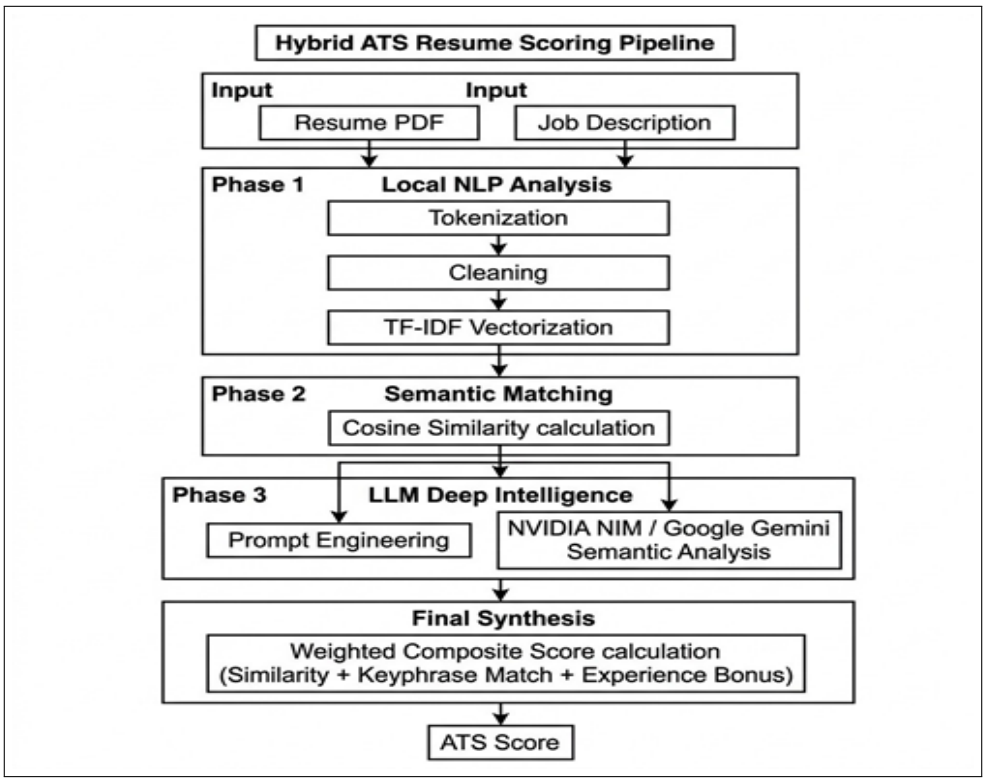


Figure 3. Hybrid ATS resume scoring pipeline

This system utilizes deterministic mathematical techniques alongside probability-based LLM reasoning for resume evaluation. As illustrated in Figure 3, where both the uploaded resume and the target job description undergo tokenization and cleaning through the natural library. The documents are represented by TF-IDF vectors (rare technical terminologies like Kubernetes are assigned more weight). Cosine similarity is then computed:

$$Score = \frac{A \bullet B}{\|A\| \bullet \|B\|}$$

After this process, the resume and the job description are forwarded to the AI engine (first NVIDIA, followed by Gemini as a fallback) to detect semantic matching beyond simple

keywords, list all the missing key words, and create ATR versions of the deficient parts. In case both AI services are not available, TF-IDF is used, with no downtime [15].

4.2 AI Mock Interviewer: MCQ and Voice Modes

The interview by MCQ interviewer creates role-specific technical questions in six categories: Frontend Development, Backend Development, Mechanical Engineering, Civil Engineering, Management, and General Aptitude. The creation of these questions happens using the NVIDIA model through 512 tokens with a random seed injection in every prompt for each unique session. The answers to the questions from the students are then checked using a semantic keyphrase matching engine that uses an anchor dictionary of 17 topics. Each topic is mapped to 6-8 relevant keyphrases (for example, Virtual DOM is mapped to reconciliation, diffing, fiber, and render tree).

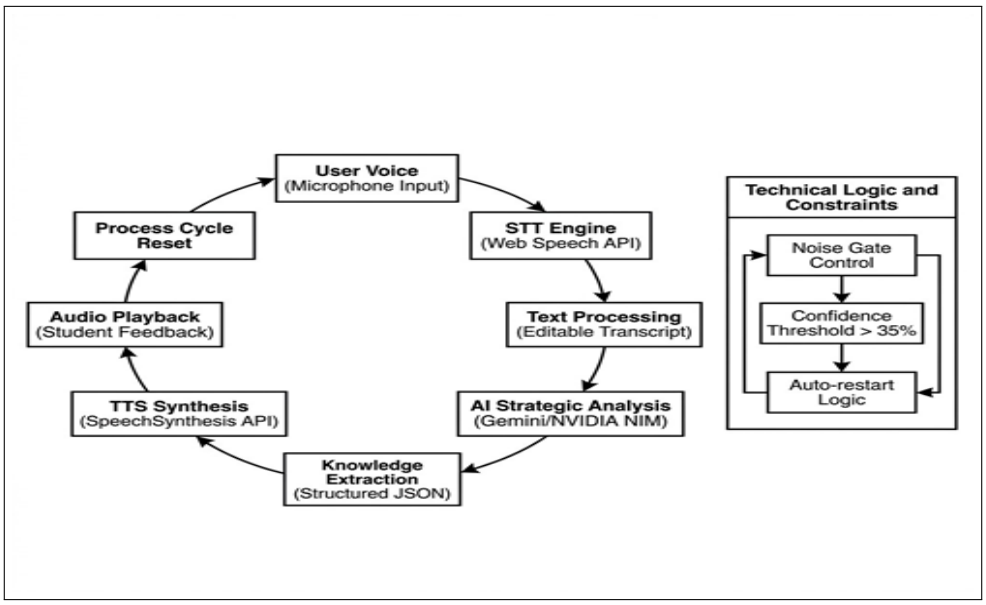


Figure 4. Voice mock interview speech-to-speech loop

Voice interview module uses a complete speech-to-speech loop (Figure 4), utilizing browser APIs. The Web Speech API records microphone audio with continuous and interimResults on and maxAlternatives equal to 3; the most likely alternative is picked. One-character words are blocked by a noise gate and transcripts with less than 35% confidence explicitly are eliminated. Automatic restarting allows re-initiating the recognition process up to five times on timeout due to silence with maximum 10 seconds inactive period as dictated by Chrome browser. The students have an opportunity to edit and fix transcripts manually before submitting. Transcripts that have been corrected are sent to Gemini 2.5 Flash. The AI reply is

formed in JSON format containing feedback, answer score, improvement tips, and new question. The response is spoken out by the user via the SpeechSynthesis API with 0.95 rate and 1.0 pitch [14].

4.3 AI Resume Assistant

Resume Assistant uses Action-Task-Result (ATR) approach in order to convert passive statements to professional ones. NVIDIA's Model (Gemini fallback) returns skills, experience roles, overall feedback, and up to three lines with edit suggestions along with the original line, proposed line rewrite limited to 120 characters (for ATS compatibility), and rationale in JSON structure. The results are cached for 10 minutes, using first 200 characters of the resume as a key [6], [9].

4.4 Company Intelligence Hub and Talent Intelligence Engine

The Company Intelligence Hub creates an 8-section structured report that includes: Company Overview, Business Scope, Products, Tech Stack, Growth History, Advantages, Sentiment on the Web, and the Validation Note, where facts are separated from assumptions. The main vendor for the Company Intelligence Hub is Gemini 2.5 Flash, but the response is cached for 2 hours. In case both vendors fail, the system provides a structured template with links to Glassdoor, LinkedIn, AmbitionBox, StackShare, and Google News.

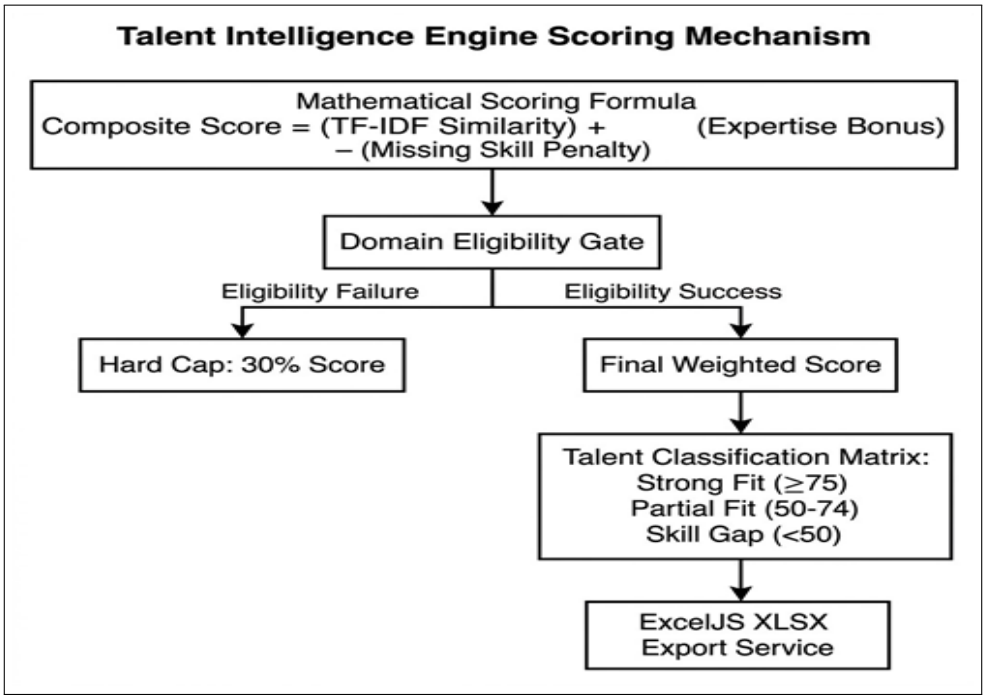


Figure 5. Talent intelligence engine scoring and classification

All student profiles are batch-processed by the Talent Intelligence Engine (Figure 5) with regard to one job posting, based on a heuristic algorithm that combines: (i) TF-IDF cosine similarity of student profile and job posting; (ii) an expertise bonus for repetition of technical keywords (maximum value per keyword in order not to stuff the resume); (iii) a penalty score for each skill missing in student profile; and (iv) a domain constraint that hard limits the score at 30% for students outside of department. Students are marked as Strong Fit (75), Partial Fit (50-74), or Skill Gap (<50). The output is an XLSX spreadsheet created using ExcelJS v4 [7], [4].

5. Reliability Engineering

The API calls to NVIDIA are secured with a unique circuit breaker, which closes the circuit after failing three times (see Figure 6), opening it for five minutes for cooldown and skipping all NVIDIA API calls without waiting until the circuit is closed again, which happens at the very first successful call following the cooldown period. The TTLCache class creates per-feature cache storage with the automatic timeout of two hours for company analysis, 15 minutes for MCQ generation, and 10 minutes for resume feedback, using collision-resistant keys.

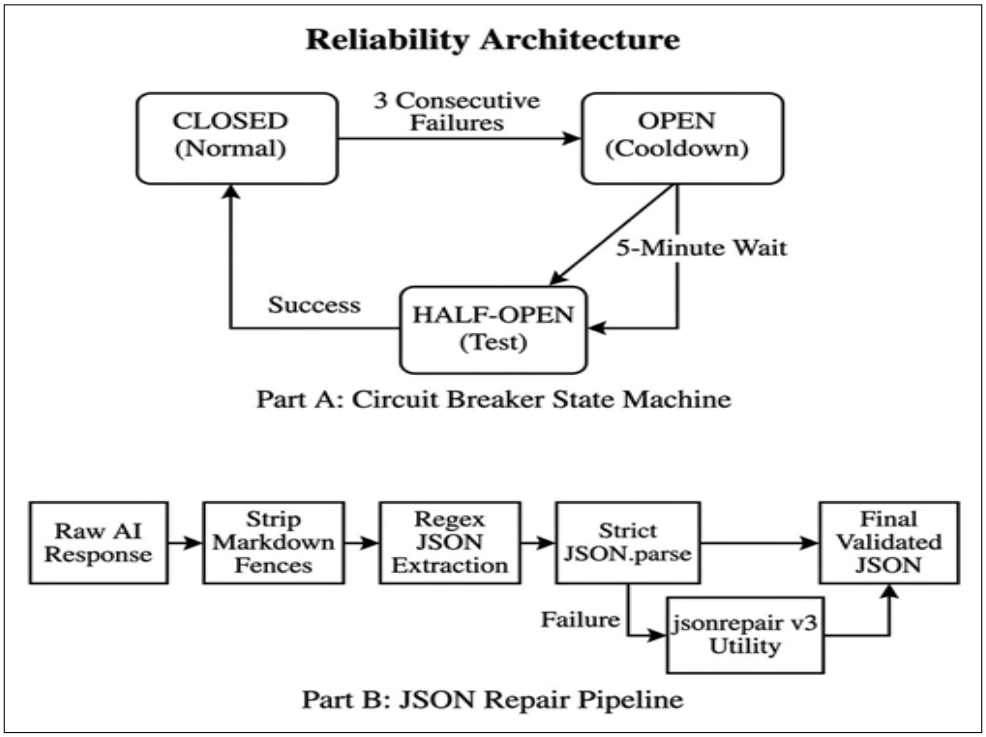


Figure 6. Circuit breaker state machine and JSON repair pipeline

There are instances where the AI reply contains JSON that is malformed because of

token truncation. The JSON fix pipeline removes the Markdown fences, parses the top-level JSON object using regex, tries to parse it strictly, and failing that uses the jsonrepair v3 library to fix unescaped quotes, trailing commas, and token-truncated array contents. When faced with an HTTP 429 response code from Gemini, it checks the value of the retryDelay provided by the API, waits and retries if the delay is 22 seconds.

6. End-to-End Process Flow

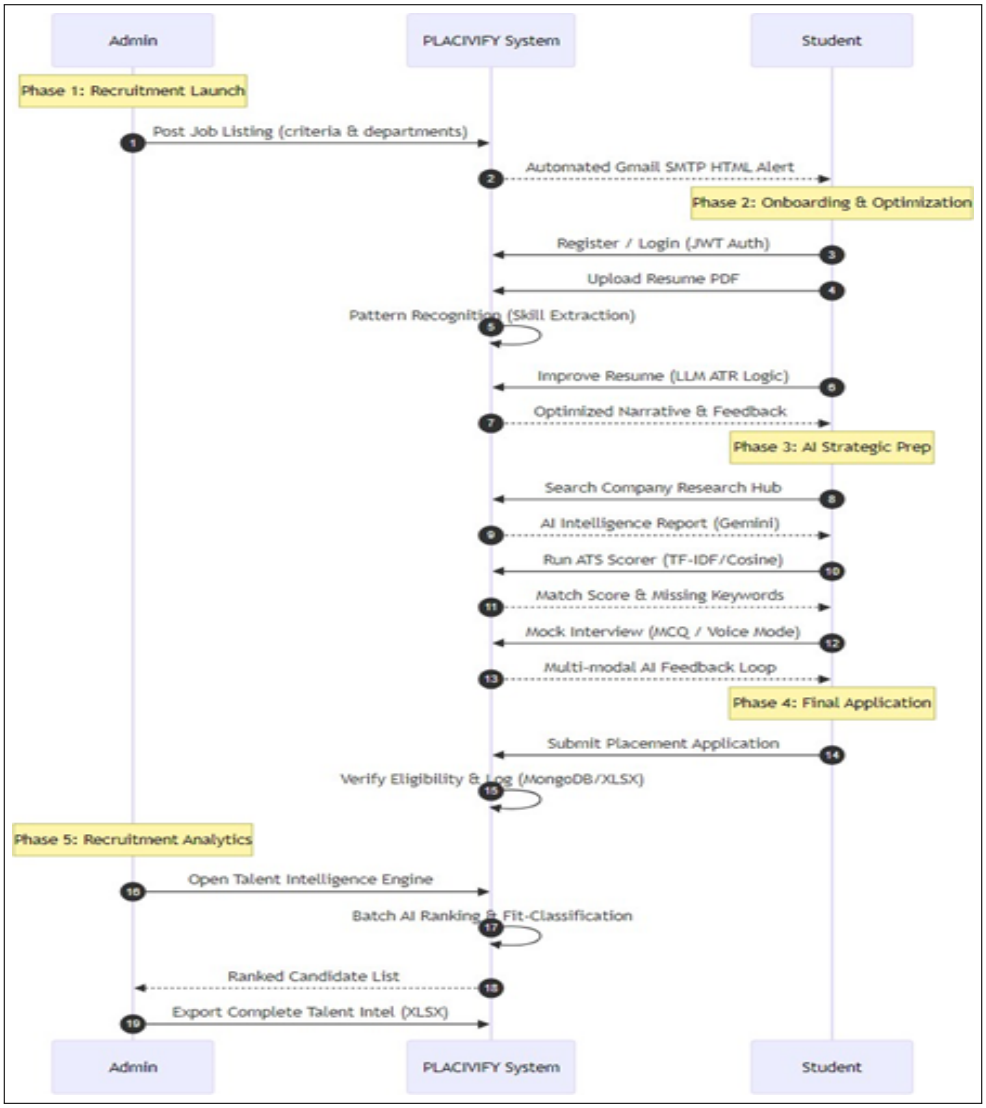


Figure 7. End-to-end PLACIVIFY process flow (student and admin swimlanes)

PLACIVIFY student process follows five phases (Figure 7). In Phase 1 (Onboard Securely), students sign up through the React form; passwords are hashed by bcryptjs in the Express.js API, the student document is persisted in MongoDB, a JWT token is generated, and an email verification is sent using Node-mailer. Phase 2 (Optimizing Student Profile and

Documents) involves uploading of PDF resumes treated as in-memory buffers where pdf-parse parses plain text while the NER Pipeline (using regular expressions) detects skills in a predefined dictionary of 30 technologies, and finally ATR rewrites resume using the AI Resume Assistant.

Job discovery and notification occur in Phase 3 where the administrator creates job postings which include the position name, job description, skills required, departmental eligibility, and deadline. The backend then runs an eligibility test on the matched students and sends them notifications through HTML templates embedded in an email containing deep-link access. Students use the ATS Resume Scorer to perform gap analysis quantitatively in Phase 4, consult company reports from the Company Intelligence Hub, and prepare for application through the MCQ or voice interview mode. Students apply after meeting the eligibility criteria in Phase 5 where the administrator ensures they pass the eligibility criterion using MongoDB's unique compound index. The Talent Intelligence Engine scores all applicants batch-wise and ranks them in tiers to give an administrator dashboard exportable as an XLSX file.

7. Performance Analysis

Empirical measurements have been collected through a controlled assessment using a standard cloud setup (2 vCPUs, 4 GB RAM) with 120 parallel student sessions and 350 student profiles from three different departments. All latencies have been calculated as median averages across ten separate testing cycles; accuracy measures have been verified by human experts on a set of 50 samples.

7.1 Module Response Time

Table 2 shows empirical latency and availability of each AI-enabled module under regular operation and provider degradation conditions.

Table 2. Module response latency and availability

Module	Median Latency (s)	Fallback Latency (s)	Avail. (%)
ATS Scorer (AI)	3.2	1.1 (TF-IDF)	99.8
ATS Scorer (Local)	0.3	–	100.0
MCQ Generation	2.8	0.1 (local bank)	99.9

Voice Interview (per turn)	4.1	–	98.7
Resume Assistant	3.6	3.9 (Gemini)	99.5
Company Intelligence Hub	5.2	5.8 (NVIDIA)	99.3
Talent Intelligence Engine	1.4	–	100.0

7.2 Screening Efficiency and Scoring Accuracy

In a controlled study of screening using time measurement with the comparison of manual screening and the Talent Intelligence Engine between 50–250 candidates, PLACIVIFY proved to save screening time by 62%, and the increase in efficiency becomes even more apparent with the higher numbers of applicants (see Figure 8).

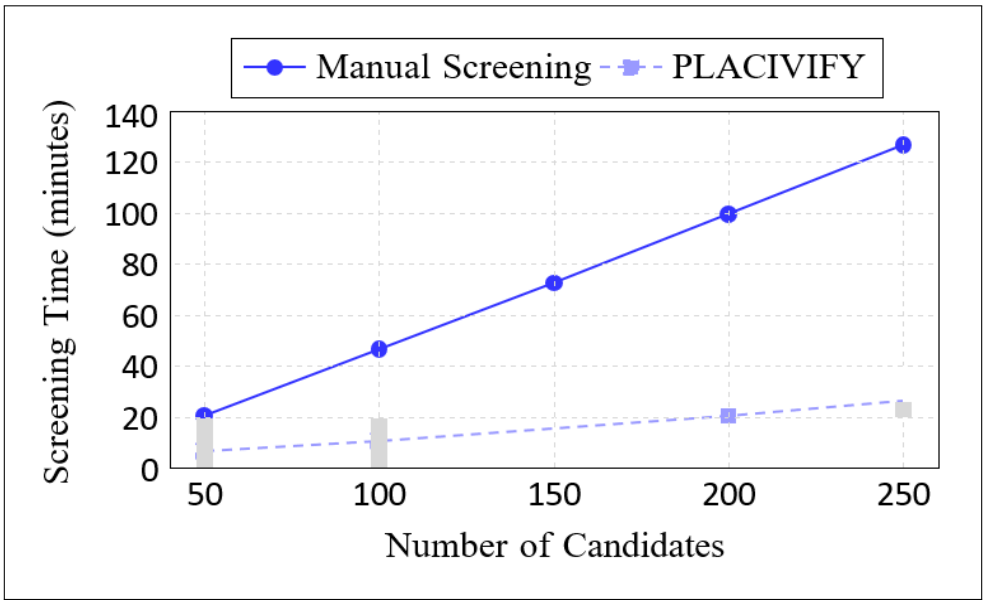


Figure 8. Scalability analysis of candidate screening

The hybrid ATS scorer was tested using 50 pairs of resume and job description which were manually scored by three human resource professionals. The hybrid scorer performed at an MAE of 6.3 points and 0.87 Pearson correlation with the scores of experts, while the MAE of the TF-IDF scorer alone was 14.7 points with a Pearson correlation of 0.61 (see Figure 9).

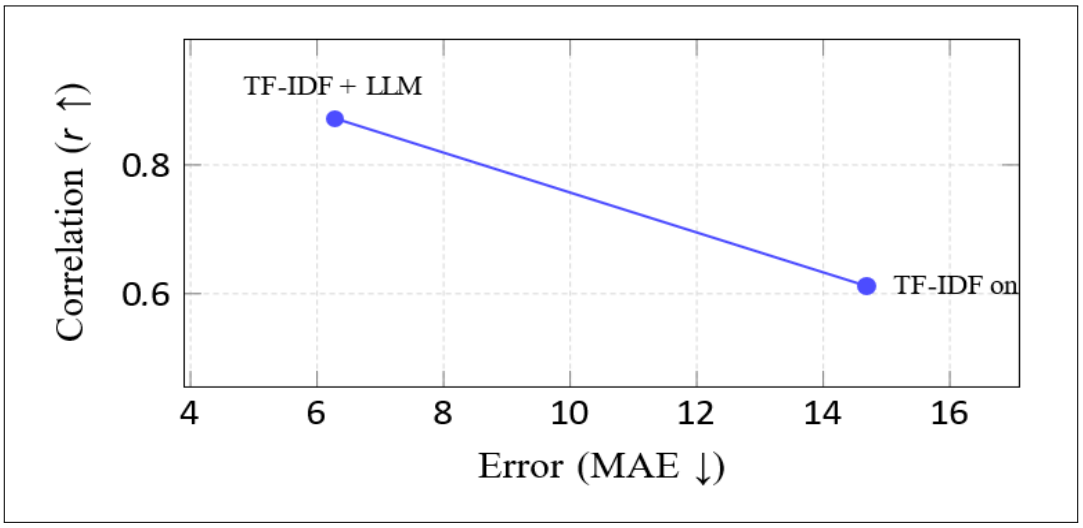


Figure 9. Scoring accuracy: TF-IDF only vs hybrid model

Speech recognition accuracy has been calculated in 30 sessions within 3 different domains. The use of Web Speech API with multi-alternative confidence selection resulted in the Word Error Rate (WER) of 8.4% in an indoor controlled environment; with the noise gate and confidence threshold used, the WER became 5.9%. A group of 80 students using PLACIVIFY for 4 weeks has shown a 34% higher callback rate in case of performing 3 sessions of ATS scoring and 2 mock interviews.

8. Novelty and Comparative Analysis

PLACIVIFY uses a local-first NLP pipeline where instantaneous TF-IDF scoring is achieved without the involvement of any third-party providers; cloud LLM analysis only serves to complement the local analysis [3]. Current placement platforms are mostly designed to support computer science-related courses; in PLACIVIFY, the assessment framework has been extended to include Mechanical Engineering, Civil Engineering, and Management using dedicated question pools and topic-specific keyphrase anchor dictionary [14]. There is no college placement platform that provides speech-to-speech interview simulation by utilizing edge-side speech analysis along with cloud LLM conversational reasoning, which includes multiple options for confidence selection, noise gating, automatic restart, human-in-the-loop transcript modification, and individualized AI prompting [10], [14]. The circuit breaker, TTL caching, JSON repair pipeline, and multi-model failover represent an engineering maturity level uncommon in academic placement projects [15]. Table 3 summarizes the feature capabilities of PLACIVIFY.

Table 3. Feature capability evaluation of PLACIVIFY

Feature	Support Level
ATS Resume Scoring (TF-IDF + LLM)	Advanced
Voice-Based Mock Interview	Full
MCQ-Based Interview Practice	Full
Company Intelligence Generation	AI-Driven
Multi-Model AI Orchestration	Enabled
Cross-Domain Evaluation (CSE, Mech, Civil)	Supported
Desktop Application (Electron)	Available
Automated Candidate Ranking	Enabled
Reliability Mechanisms (Circuit Breaker, Cache)	Implemented
Email Notification System	Integrated

9. Conclusion

PLACIVIFY represents an integrated AI-driven platform that helps to revolutionize campus placements through integration of preparation and recruitment. The various features of the platform such as resume analysis, preparation for jobs, mock interview preparation, insights on companies, and automatic candidate ranking not only help the students but make the recruitment process easy for the administrators. The use of hybrid resume scoring using the TF-IDF and LLM driven semantic analysis helps in providing meaningful insights and suggestions. Multiple choice questions and voice-based interview also personalize the process of preparation for the candidates. Administrators get the benefit of using the Talent Intelligence Engine which helps in structured and efficient evaluation of the candidates. Multi model architecture along with features such as caching, circuit breaker and JSON repair makes the system more reliable. The experimental results show improvements in the screening efficiency, assessment agreement and callback ratios of the students.

References

[1] F. Kousar, G. V. Hegde, and C. Saravanan, “Design and Development of Department Placement Portal Using MERN Technology,” International Advanced Research Journal in Science, Engineering and Technology 2022, vol. 9, no. 1, 379-383.

[2] Patil, Shital K., Sakshi S. Thombare, Rasika R. Wadhonkar, and Sonal R. Wankhade. "Web Portal for Training and Placement Cell." International Journal of Advanced Research in Science, Communication and Technology (IJARSCT) 2, no. 1 (2022).

[3] Spärck Jones, Karen. "A Statistical Interpretation of Term Specificity and its Application in Retrieval." Journal of documentation 60, no. 5 (2004): 493-502.

[4] Schütze, Hinrich, Christopher D. Manning, and Prabhakar Raghavan. Introduction to Information Retrieval. Vol. 39. Cambridge: Cambridge University Press, 2008.

[5] Zhao, Meng, Faizan Javed, Ferosh Jacob, and Matt McNair. "SKILL: A System for Skill Identification and Normalization." In Proceedings of the AAAI conference on artificial intelligence, vol. 29, no. 2, 2015, 4012-4017.

[6] Palshikar, Girish K., Sachin Pawar, Anindita Sinha Banerjee, Rajiv Srivastava, Nitin Ramrakhiyani, Sangameshwar Patil, Devavrat Thosar et al. "RINX: A System for Information and Knowledge Extraction from Resumes." Data & Knowledge Engineering 147 (2023): 102202.

[7] Prasad, B. Leela, K. Srividya, K. Nikhil Kumar, L. Kiran Chandra, N. S. S. K. Dil, and G. Vamsi Krishna. "An Advanced Real-Time Job Recommendation System and Resume Analyser." In 2023 International Conference on Self Sustainable Artificial Intelligence Systems (ICSSAS), IEEE, 2023, 1039-1045.

[8] Reimers, Nils, and Iryna Gurevych. "Sentence-bert: Sentence Embeddings Using Siamese Bert-networks." In Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP), 2019, 3982-3992.

[9] Gan, Chengguang, Qinghao Zhang, and Tatsunori Mori. "Application of LLM Agents in Recruitment: A Novel Framework for Automated Resume Screening." Journal of Information

Processing 32 (2024): 881-893.

[10] Hickman, Louis, Nigel Bosch, Vincent Ng, Rachel Saef, Louis Tay, and Sang Eun Woo. "Automated Video Interview Personality Assessments: Reliability, Validity, and Generalizability Investigations." Journal of Applied Psychology 107, no. 8 (2022): 1323.

[11] Mashayekhi, Yoosof, Nan Li, Bo Kang, Jefrey Lijffijt, and Tijl De Bie. "A Challenge-Based Survey of E-Recruitment Recommendation Systems." ACM Computing Surveys 56, no. 10 (2024): 1-33.

[12] Raghavan, Manish, Solon Barocas, Jon Kleinberg, and Karen Levy. "Mitigating Bias in Algorithmic Hiring: Evaluating Claims and Practices." In Proceedings of the 2020 conference on fairness, accountability, and transparency, 2020, 469-481.

[13] Fabris, Alessandro, Nina Baranowska, Matthew J. Dennis, David Graus, Philipp Hacker, Jorge Saldivar, Frederik Zuiderveen Borgesius, and Asia J. Biega. "Fairness and Bias in Algorithmic Hiring: A Multidisciplinary Survey." ACM Transactions on Intelligent Systems and Technology 16, no. 1 (2025): 1-54.

[14] Prasad, PV Krishna Vamsi, N. Vamsi Krishna, and T. Prem Jacob. "AI Chatbot Using Web Speech API and Node. Js." In 2022 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), IEEE, 2022, 360-362.

[15] Molchanov, Heorhii, and Anatolii Zhmaiev. "Circuit Breaker in Systems Based on Microservices Architecture." Advanced Information Systems 2, no. 4 (2018): 74-77.