iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** X **Month of publication:** October 2026

DOI: https://doi.org/10.22214/ijraset.2026.85283

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue X Oct 2026- Available at www.ijraset.com

PrepAI: A Web Platform for Placement Preparation with Aptitude Tests, Online Coding Practice, Mistake Review and AI-Generated Study Plans

Sudhanshu Bhaurkar[1], Aasim Ansari[2], Shlok Kotpalliwar[3], Dr. Amita Meshram[4]
[1, 2, 3]*Department of Computer Science and Engineering, G H Raisoni University, Amravati, Maharashtra, India | Session 2026-27*
[4]*Assistant Professor, CSE*

Abstract: ***In today's challenging job market, effective placement preparation has taken on increased significance, driven by economic instability and heightened instances of layoffs. This makes securing a suitable placement more critical than ever for students and job seekers. However, existing resources often fall short in providing the necessary support and materials. PrepAI is a web-based placement preparation platform that brings aptitude practice, coding practice and AI-generated career guidance into one place. It offers timed aptitude tests, a coding workspace that runs the student's code against test cases, a Mistake Notebook that collects failed attempts, a dashboard with progress figures, and JobFit AI, which turns a job description into a day-by-day study plan. The front end uses React with Vite, the back end uses Node.js and Express with JWT login, and the data sits in a PostgreSQL database on Neon. Code is run through Judge0, and the Groq API produces the AI feedback and study plans. This paper explains the design, describes how each module works, and presents the results of the running prototype with screenshots. No study with other students has been run yet, so no measured improvement in scores is claimed.***
Keywords: PrepAI, placement, aptitude test, coding judge, LLM, career planning.

I. INTRODUCTION

Most engineering students face the same sequence in placements: an online aptitude test, a coding round, and then technical and HR interviews. Doing well needs steady practice, and it also needs some idea of which topics are weak.

The usual way to prepare is to mix several tools. A student might solve aptitude questions on one website, practise coding on another and follow a generic roadmap from a video. Each tool does its own job, but they do not share any data. A student who keeps failing questions on stacks, for example, will not see that pattern unless they keep notes by hand. Generic roadmaps also ignore the specific role a student is applying for. As a result, a student preparing for placements has no single tool that tests aptitude and coding skills, explains mistakes, shows progress across both, and turns a particular job description into a study plan that fits the time left. The existing resources are scattered and generic.

Large language models (LLMs) have made some of this easier. A model can read a job description and list the skills it asks for, and it can comment on a piece of code. Code-execution services such as Judge0 let a platform check a solution against test cases without running unknown code on its own server. These two ideas are combined here with an ordinary web application and a shared database. The work in this paper has four parts. First, one platform that holds aptitude practice, coding practice, mistake review, progress tracking and career planning. Second, a coding workspace that runs code against test cases and then offers AI feedback. Third, a Mistake Notebook that gathers failed attempts and lets students filter them by title, difficulty and language. Fourth, JobFit AI, which takes a job description, the student's current skills and the number of days available, and returns a skill-gap list, a daily plan, resources and interview topics.

Section II reviews related work, Section III describes the design, Section IV covers implementation, Section V shows results and Section VI concludes.

II. RELATED WORK

1) M. Saatçı, R. Kaya and R. Ünlü. NLP-based competency extraction with Jaccard and cosine similarity can match resumes to jobs automatically in a fair, bias-reduced way.

©IJRASET: All Rights are Reserved

2) F. P. W. Lo, J. Qiu, Z. Wang, H. Yu, Y. Chen, G. Zhang and B. Lo. A multi-agent LLM pipeline of extractor, evaluator and summarizer gives explainable, role-specific resume screening.
3) C. Kim, J. Choi, J. Yoon, D. Yoo and W. Lee. Combining video, audio and text in a fairness-aware model improves accuracy and reduces demographic bias in interview scoring.
4) R. V. K. Bevara, N. R. Mannuru, S. P. Karedla et al. Transformer embeddings (BERT, RoBERTa, GPT, Gemini, Llama) rank candidates up to 15.9% better than keyword-based ATS.
5) E. Q. Tseng, P. C. Huang, C. Hsu, P. Y. Wu, C. T. Ku and Y. Kang. The CodEv framework grades code with an LLM and gives rubric-aligned scores and constructive feedback on student submissions.
6) C. N. Hang, C. W. Tan and P. D. Yu. MCQGen generates and auto-validates diverse, difficulty-calibrated multiple-choice questions for personalised aptitude practice.
7) J. Yu, Y. Zhuang, Y. Sun, W. Gao, Q. Liu, M. Cheng et al. TestAgent, a conversational LLM agent for adaptive testing, reaches accurate results with 20% fewer questions than standard methods.
8) P. C. Siswipraptini, H. L. H. S. Warnars, A. Ramadhan and W. Budiharto. A machine-learning model can recommend career paths by matching IT students to suitable job specializations from their skill profiles.

III. SYSTEM DESIGN

A. Overview and architecture

A visitor opens the landing page, signs up or logs in, and arrives at the dashboard. From there they can open the Aptitude Test, Coding IDE, Mistake Notebook, JobFit AI or About pages. The system has three layers. The front end is a React single-page application built with Vite. The back end is a Node.js and Express API with routes for authentication, questions, submissions, dashboard data and AI requests. The database is PostgreSQL on Neon and holds users, questions and submissions. The back end also talks to two outside services: Judge0 for running code and the Groq API for AI feedback and study plans.

B. Login and protected pages

After a successful login the server returns a JSON Web Token (JWT). The front end sends it with later requests, and the server checks it before returning protected data. On the client, protected routes send anyone who is not logged in back to the login page.

C. Aptitude test

The test pulls 25 multiple-choice questions at random from the database for a chosen subject, such as Quantitative Aptitude or Logical Reasoning. A countdown timer runs during the test. Each question has four options. A navigator panel shows how many questions are answered, how many remain, and a grid for jumping to any question. Answers are saved as the student moves along. When the student submits, the server scores the test, stores the attempt and shows the score with the correct answers. The AI feedback then summarises strengths, weaknesses and suggestions.

D. Coding workspace

The student picks a problem from a list. The problem page shows the description, examples, constraints, topics and hints. The student writes a solution in a Monaco-based editor in one of four languages. The code is sent to the back end, which passes it to Judge0 together with the test cases and reads back the results. The student sees which tests passed and what the program printed. An AI Feedback tab analyses the submitted code and suggests improvements. The design separates a Run action that uses only the visible test cases from a Submit action that also uses hidden test cases. The present build shows a single Run and Submit button, as seen in Fig. 5.

E. Mistake Notebook and dashboard

Every failed attempt is stored with its result. The Mistake Notebook reads these records and shows three numbers: total failed coding submissions, unique questions answered wrongly, and failed test cases. Students can search by title and filter by difficulty and language, then reopen an old attempt to try again. The dashboard reads the same records and shows the number of practice submissions, successful submissions, the coding pass rate, a skill matrix per subject and a Focus Areas list of topics to revisit.

F. JobFit AI

JobFit AI needs three inputs: the job description (at least 100 and at most 15,000 characters), the skills the student already has,

©IJRASET: All Rights are Reserved

separated by commas, and the preparation time, for example 7 days. The back end sends these to the Groq API and gets back a structured plan. The page shows it under five tabs: Overview, Skill Gaps, Daily Plan, Resources and Interview Prep. The overview includes the role summary, counts of required skills, study days, resources and tasks, a progress bar, the priority skills ranked by importance, a short preview of the daily plan and a list of interview topics.

G. Database

Table group	What it stores
Users	Account details used for login
Questions	Aptitude questions with options and answers; coding problems with descriptions, examples, constraints, hints and test cases
Submissions	Each attempt and its result: aptitude scores and answers, submitted code, language, status and test outcomes

The dashboard, the Mistake Notebook and the progress figures are all computed from the submissions data. This is the main reason for using one shared database for all modules.

IV. IMPLEMENTATION

Layer	Technology	Used for
Front end	React with Vite	Pages, routing, interface
Back end	Node.js with Express	REST API and calls to outside services
Database	PostgreSQL on Neon	Users, questions, submissions
Login	JSON Web Tokens	Authentication and protected routes
Code execution	Judge0	Running Python, JavaScript, C++ and Java code
AI	Groq API	Code feedback, aptitude feedback and JobFit AI plans

A coding submission follows these steps. The front end sends the code, the language and the problem ID to the back end along with the token. The server checks the token, loads the test cases for that problem, sends the code to Judge0 and collects the outputs. It compares them with the expected outputs, decides whether the attempt passed, saves the submission and returns the per-test results. Aptitude submissions follow the same path, except that the server scores the answers itself. Running student code on Judge0 means the application server never executes it. The interface has a dark theme and a fixed sidebar with Dashboard, Aptitude Test, Coding IDE, About Us, Mistake Notebook and JobFit AI. The logged-in email and a Log out link sit at the bottom of the sidebar.

V. RESULTS AND DISCUSSION

The platform runs as a working prototype. This section shows screenshots of each module and explains what they show. The screenshots were taken from a local development server with a sample account. They show that the modules work together, but they are not a user study.

A. Screenshots of the working system

Landing page. Fig. 1 is the page a new visitor sees. It has the PrepAI name, Login and Get Started buttons, the line "Prepare Smarter. Perform Better.", and an About section. Logging in takes the user to the dashboard.

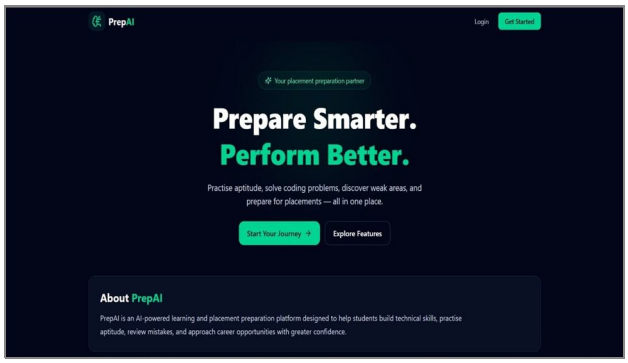


Fig. 1. PrepAI landing page.

Dashboard. Fig. 2 shows the dashboard for the sample user: 5 practice submissions, 2 successful ones and a 50% coding pass rate (2 of 4 coding attempts). The skill matrix gives Logical Reasoning 55% and Quantitative Aptitude 70%, and both topics appear under Focus Areas.

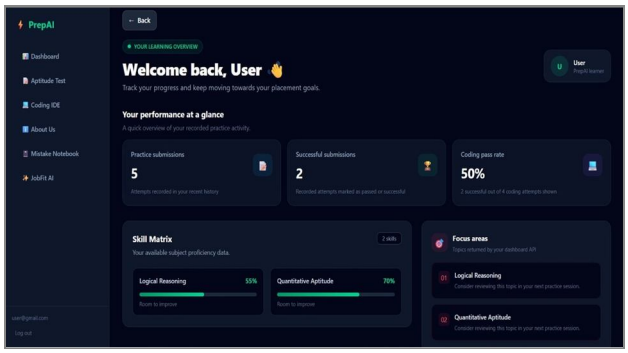


Fig. 2. Candidate dashboard with performance cards, skill matrix and focus areas.

Fig. 3 shows the Recent Submissions table on the same page. It lists the aptitude test with its percentage score and four coding attempts: Fibonacci Number (passed 2/2), Valid Parentheses (failed 0/2), Fibonacci Number (failed 0/2) and Two Sum (passed 2/2). The note below the table says these figures come from recent records only. That is why the dashboard shows fewer attempts than the Mistake Notebook.

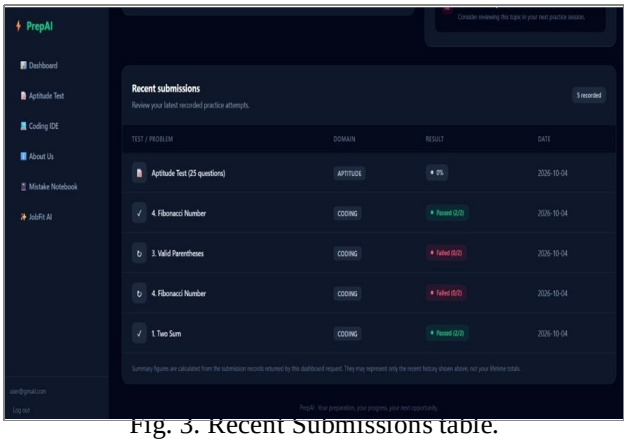


Fig. 3. Recent Submissions table.

Aptitude test. Fig. 4 shows a Quantitative Aptitude mock test at question 1 of 25 with 9:55 left on the timer. The panel on the right shows answered and remaining counts, the completion percentage, the question grid and the Submit Mock Paper button, which reports that 25 questions are still unanswered.

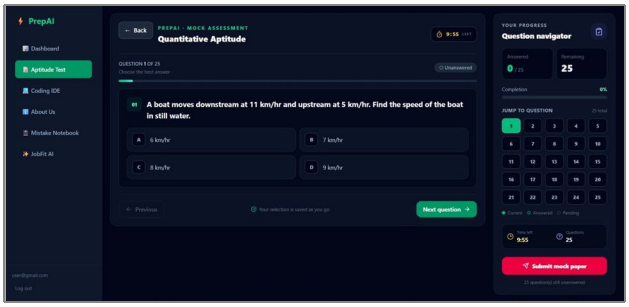


Fig. 7. JobFit AI input form.

Fig. 8 shows the plan that came back. It reports 12 required skills, 7 study days, 10 learning resources and 0 of 14 tasks done, with the progress bar at 0%. The plan is split into the tabs Overview, Skill Gaps, Daily Plan, Resources and Interview Prep.

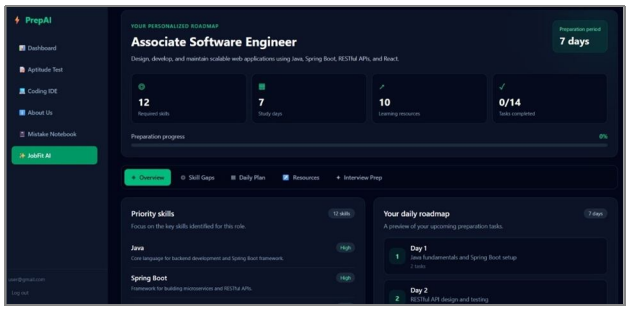


Fig. 4. Timed aptitude test with the question navigator.

Coding workspace. Fig. 5 shows the workspace with 12 problems listed on the left, tagged by topic. The selected problem is Fibonacci Number (Easy) with its description, two examples, constraints and topics. The editor on the right has the Java solution, and the result panel reports that 2 of 2 configured tests passed. The AI Feedback tab sits next to the Test Results tab.

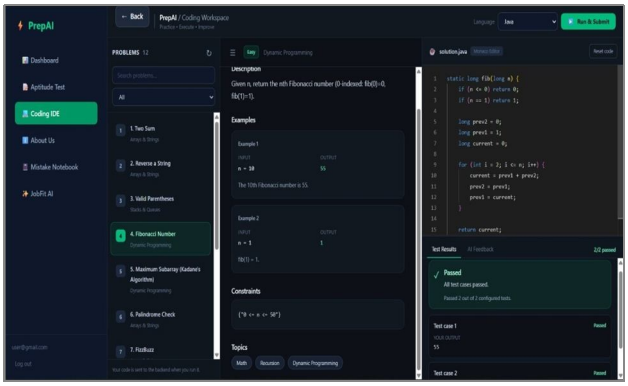


Fig. 8. Roadmap generated by JobFit AI.

Fig. 9 shows the rest of the overview. Java, Spring Boot, RESTful APIs, React and JavaScript (ES6+) are marked as high priority, each with a one-line reason. Day 1 covers Java fundamentals and Spring Boot setup, Day 2 covers REST API design and testing, and Day 3 covers React component architecture. The interview section lists object-oriented programming, data structures and algorithms, REST API design principles and Spring Boot core features.

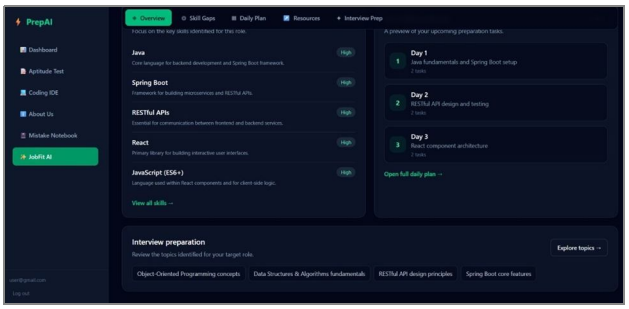


Fig. 5. Coding workspace with problem, editor and test results.

Mistake Notebook. Fig. 6 shows 12 failed coding submissions, 3 unique questions answered wrongly and 24 failed test cases. The search box and the difficulty and language filters act on the 12 stored submissions, and the Review and Improve list below lets the student reopen each one.

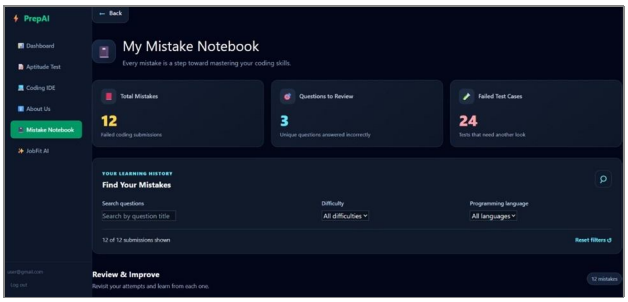


Fig. 6. Mistake Notebook.

JobFit AI. Fig. 7 shows the input form. An Associate Software Engineer job description (1,093 of 15,000 characters) was pasted, the skills Java, SQL, React, HTML, CSS, JavaScript and Git were entered, and 7 days was chosen.

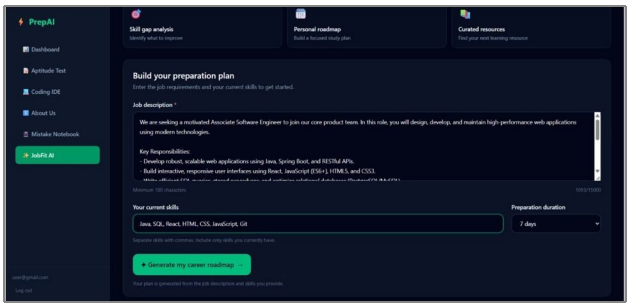


Fig. 9. Priority skills, daily plan and interview topics.

B. Discussion

The main benefit seen in the prototype is that the modules share data. A failed coding attempt shows up in the Mistake Notebook, lowers the pass rate on the dashboard and can point to a topic under Focus Areas without the student doing anything. JobFit AI ties the plan to a real job description instead of a general topic list. The papers reviewed [1]-[8] mostly serve recruiters or handle one task, so PrepAI can be seen as a student-side tool that joins practice, evaluation, review and planning. The table below summarises the difference from using separate tools.

Aspect	Separate tools	PrepAI
Aptitude and coding practice	Different sites and accounts	One platform, one account
Code checking	Often self-checked	Executed against test cases through Judge0
Feedback	Generic or none	AI feedback on code and aptitude results

Aspect	Separate tools	PrepAI
Mistake review	Handwritten notes	Mistake Notebook with search and filters
Progress	Not combined	Dashboard with skill matrix and focus areas
Career planning	Generic roadmaps	Plan built from a given job description and time

VI. CONCLUSION AND FUTURE WORK

PrepAI is a single web platform for placement practice. It combines timed aptitude tests, a coding workspace backed by Judge0, a Mistake Notebook, a dashboard and the JobFit AI planner. Because every attempt goes into one database, the platform can show a student where they are weak and suggest a plan for a specific job. The screenshots in Section VI show that each module works in the prototype.

The next steps are:

1) test the platform with a group of students and measure changes in aptitude scores and coding pass rates;
2) grow the question bank and add adaptive question selection, as in [6] and [7]; and
3) add mock interviews and resume analysis, keeping the fairness concerns in [2]-[4] in mind.

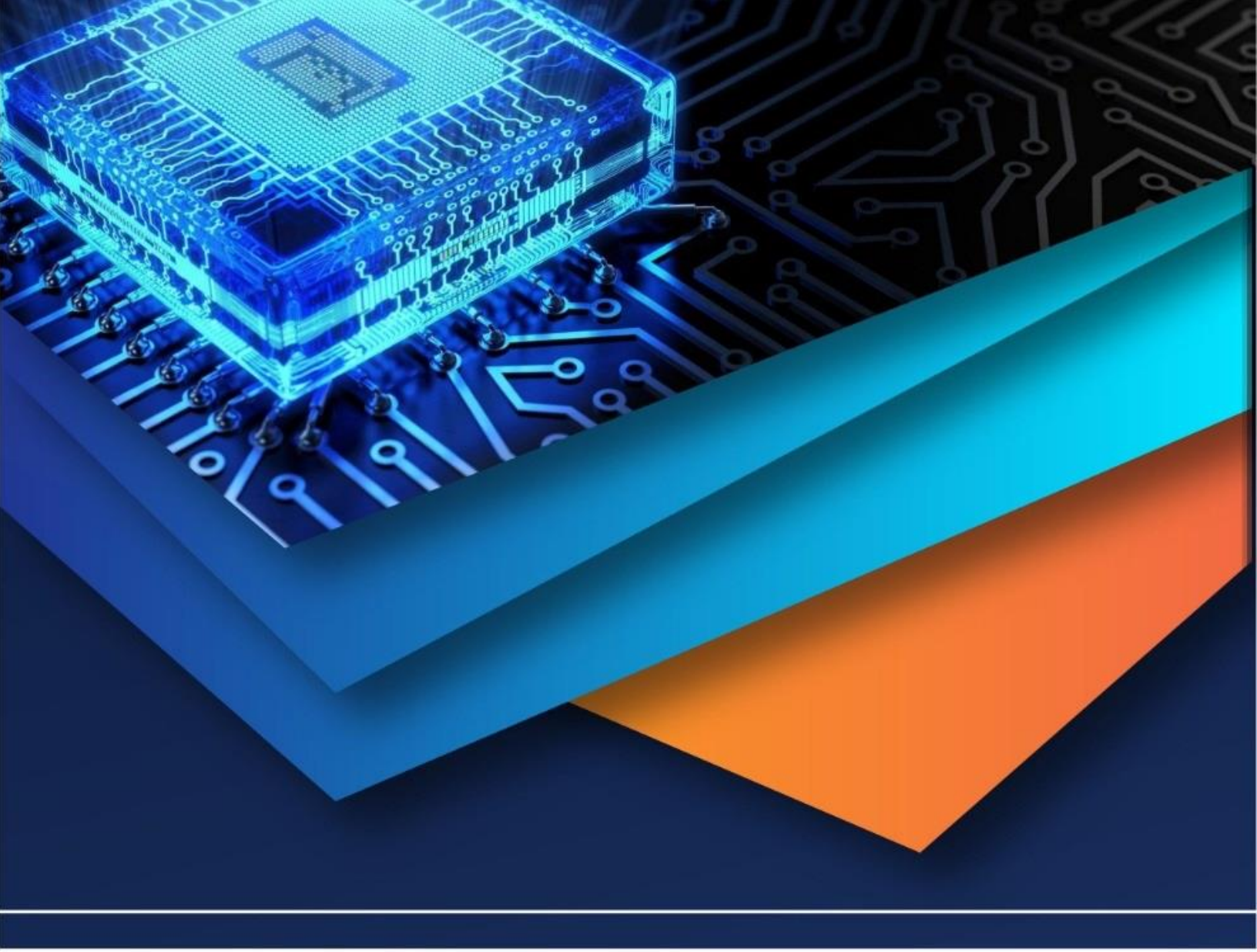
VII. ACKNOWLEDGMENT

The authors thank Dr. Amita Meshram, Assistant Professor, CSE, and the Department of Computer Science and Engineering, G H Raisoni University, Amravati, for their guidance and support.

REFERENCES

[1] Saatçı, M., Kaya, R., & Ünlü, R. (2024). Resume screening with natural language processing (NLP). Alphanumeric Journal, 12(2), 121-140.
[2] Lo, F. P. W., Qiu, J., Wang, Z., Yu, H., Chen, Y., Zhang, G., & Lo, B. (2025, June). AI hiring with LLMs: A context-aware and explainable multi-agent framework for resume screening. In 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) (pp. 4184-4193). IEEE.
[3] Kim, C., Choi, J., Yoon, J., Yoo, D., & Lee, W. (2023). Fairness-aware multimodal learning in automatic video interview assessment. IEEE Access, 11, 122677-122693.
[4] Bevara, R. V. K., Mannuru, N. R., Karedla, S. P., Lund, B., Xiao, T., Pasem, H., ... & Rupeshkumar, S. (2025). Resume2Vec: Transforming applicant tracking systems with intelligent resume embeddings for precise candidate matching. Electronics, 14(4), 794.
[5] Tseng, E. Q., Huang, P. C., Hsu, C., Wu, P. Y., Ku, C. T., & Kang, Y. (2024, December). Codev: An automated grading framework leveraging large language models for consistent and constructive feedback. In 2024 IEEE International Conference on Big Data (BigData) (pp. 5442-5449). IEEE.
[6] Hang, C. N., Tan, C. W., & Yu, P. D. (2024). MCQGen: A large language model-driven MCQ generator for personalized learning. IEEE Access, 12, 102261-102273.
[7] Yu, J., Zhuang, Y., Sun, Y., Gao, W., Liu, Q., Cheng, M., ... & Chen, E. (2025, July). TestAgent: An adaptive and intelligent expert for human assessment. In Findings of the Association for Computational Linguistics: ACL 2025 (pp. 724-747).
[8] Siswipraptini, P. C., Warnars, H. L. H. S., Ramadhan, A., & Budiharto, W. (2024). Personalized career-path recommendation model for information technology students in Indonesia. IEEE Access, 12, 49092-49105.

©IJRASET: All Rights are Reserved



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)