RESEARCH ARTICLE

RSP Science Hub

International Research Journal on Advanced Science Hub
2582-4376
www.rspsciencehub.com
Vol. 08, Issue 10 October

Check for updates

http://dx.doi.org/10.47392/IRJASH.2026.044

Hire Mind: An Intelligent Placement Management Platform with Machine Learning-Based Company Prioritization, Skill Gap and Resume Analysis

Jayapriya R[1], Karunya S[2], Logeshwaran K[3], Sowmiya G[4]

[1,2,3]Department of Computer Science and Engineering, Erode Sengunthar Engineering College, Erode, Tamil Nadu, India

[4]Assistant Professor, Department of Computer Science and Engineering, Erode Sengunthar Engineering College, Erode, India

Emails: *jayapriya0805@gmail.com[1], karu.s.unya@gmail.com[2], logeshwarancs25@gmail.com[3], sowmiyame22@gmail.com[4]*

Article history

Received: 05 August 2026
Accepted: 03 September 2026
Published: 07 October 2026

Keywords:
Campus Placement, Machine Learning, Company Prioritization, Skill Gap Analysis, Resume Analysis, Recommendation System, Employability Analytics.

Abstract

Campus placement requires the coordinated handling of student profiles, company drives, eligibility conditions, applications, outcomes and placement statistics. This paper presents HireMind, a role-based placement management platform that we extend with a proposed machine-learning layer to help students prepare and help staff make placement decisions. The proposed layer covers company-drive prioritization into High, Medium and Low levels based on domain and skill relevance; domain-wise skill progress and gap identification; resume strength analysis; placement preparation recommendations; and placement outcome analytics. The management prototype from the earlier paper runs on Node.js, Express and MySQL, with role-specific access and placement dashboards. We propose Python, Pandas, NumPy, Scikit-learn and FastAPI as the ML service stack, while the existing backend continues to handle application workflows and database access. The paper describes the system architecture, candidate methods, data requirements and evaluation protocol. Because the proposed ML components have not yet been implemented or experimentally validated in the supplied prototype, we make no claim about predictive accuracy or performance.

1. Introduction

Campus placement involves student academic records, technical skills, projects, certifications, internships, company requirements, eligibility rules and recruitment outcomes. When these details are kept in separate documents and communication channels, students can struggle to find the drives that suit them and to see what preparation they need. Placement staff, in turn, need a consolidated view of student information and recruitment statistics. HireMind is designed for three groups: students, placement staff and administrators. Its management layer covers student profiles, academic details, skills, projects, certifications, experience, resumes, company and role information, placement drives, eligibility details, announcements and dashboards. The proposed ML extension draws on these records to give more actionable insights, so that the platform is more than a data-entry and reporting system. The updated design centres on five connected capabilities: (i) company-drive priority

classification, (ii) domain skill progress and gap analysis, (iii) resume strength analysis, (iv) preparation recommendations based on drive requirements, and (v) placement outcome prediction and descriptive analytics. These outputs are advisory. They are meant to support student decisions and do not guarantee recruitment outcomes.

1.1.Related Work

Zhong [1] describes a university student management system that applies data mining and an artificial neural network to student information, including personalized learning and employment recommendations. The work is a useful reference for combining student management with data-driven services, although the description we had does not detail how the recommendation model was evaluated. Thilak et al. [2] examine machine-learning methods for predicting graduate placement, including Logistic Regression, Random Forest, ANN, SVM, KNN and boosting approaches. Their paper shows why datasets, validation procedures and metrics need to be reported consistently when prediction models are compared. Yadav et al. [3] present a campus job portal with student, company and administrator roles, candidate search and job recommendations based on skill keywords. Their approach supports using job requirements and student skills for matching. Sunil et al. [4] discuss a fuzzy optimization approach that uses placement-related attributes such as CGPA, aptitude, programming and communication. These studies motivate our use of structured student and job attributes in HireMind, while the present design brings together management, relevance ranking, student-facing skill insights and placement analytics.

1.2.Research Gap and Contributions

The reviewed works give examples of student management, placement prediction, job recommendation and placement-related optimization. The earlier HireMind prototype description verifies a role-based placement management layer, but the supplied code has no implemented ML pipeline. The updated proposal tackles this gap by specifying how existing placement data can be connected to separate ML services and turned into interfaces that students and staff can understand. This paper contributes: (1) an integrated architecture that connects the HireMind management backend with a separate Python ML service; (2) a domain- and skill-based mechanism for prioritizing company drives; (3) a domain progress and skill-gap view backed by evidence of demonstrated skills; (4) a resume analysis framework that reports strengths and improvement areas by category; and (5) a validation plan for recommendation, classification and outcome analytics.

2. Method

2.1.Proposed HireMind Architecture

The proposed architecture keeps the existing Node.js and Express application backend and the MySQL database. Students, placement staff and administrators use the platform through its web interface. The backend handles authentication, role-based access, profile records, drive information and placement data. A Python ML service, exposed through FastAPI, receives prepared features from the backend and returns scores, priority labels, recommendations and analysis results, which the backend then shows on the relevant dashboards.

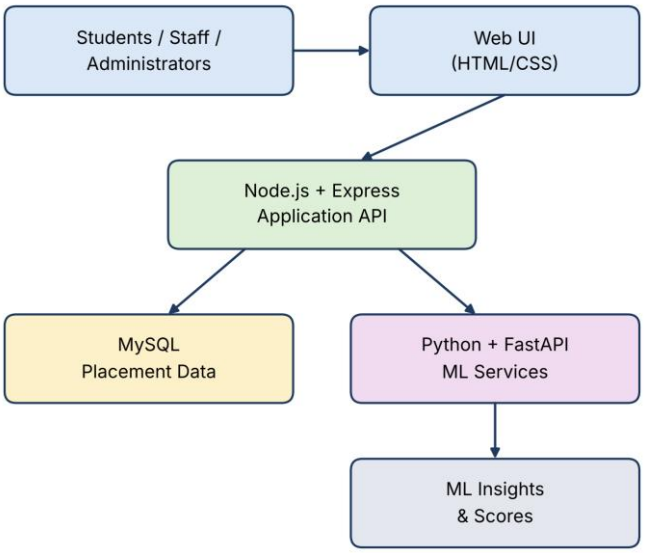


Figure 1 Proposed Hiremind Architecture with A Separate Ml Service.

2.2.Existing Management Layer and Proposed Additions

The code inspection in the earlier paper reported Node.js/Express routes, MySQL access through mysql2, bcryptjs password hashing, multer file uploads and role-specific dashboards. The schema covered student, department, batch, skill, project, certification, experience, resume, domain-interest, drive, announcement and yearly-statistics

information, and dashboard queries produced placement counts and package aggregates. The inspected code had no Python, Scikit-learn, ML endpoints, dedicated application-tracking table or notification table. We keep these findings as the baseline; the ML components and their service integration remain proposals.

Table 1 Architecture Layers, Technologies and Their Status

Layer	Technology / role	Status in this paper
Web interface	Existing HTML/CSS pages; React may be adopted for richer interactive views	Existing prototype: static pages; React is a proposed UI enhancement
Application backend	Node.js, Express, authentication and REST endpoints	Described as present in supplied prototype
Database	MySQL student, drive, skill and outcome records	Described as present in supplied schema
ML service	Python, FastAPI, Pandas, NumPy, Scikit-learn	Proposed; not found in inspected code
Resume extraction	PDF/DOCX text extraction libraries	Proposed; requires testing on uploaded formats

2.3.User-Facing ML Features

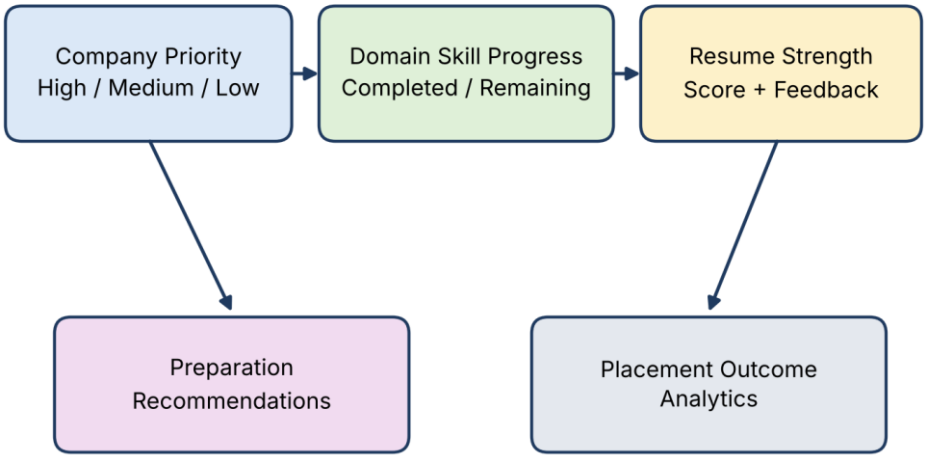


Figure 2 Proposed Intelligent Features and Their Student-Facing Outputs

Company priority shows relevant drives in High, Medium or Low categories. Domain progress displays demonstrated skills, skills still to be learned, and a percentage derived from a defined domain skill catalogue. Resume strength analysis identifies relevant skills, projects, experience and certifications and gives feedback for improvement. Preparation recommendations rank the topics or skills that appear in the requirements of target drives. Placement analytics summarizes historical outcomes by department, batch and company.

2.4.Methodology

The shared workflow starts with student-profile and company-drive data retrieved from MySQL. Preprocessing removes duplicate records, handles missing values, normalizes numeric attributes and tokenizes skill and requirement text. Feature engineering then builds student-domain, student-skill and job-requirement representations, and the ML service computes the output of each feature module. Before any model output is shown to users, it is evaluated on held-out data with metrics suited

to the task.

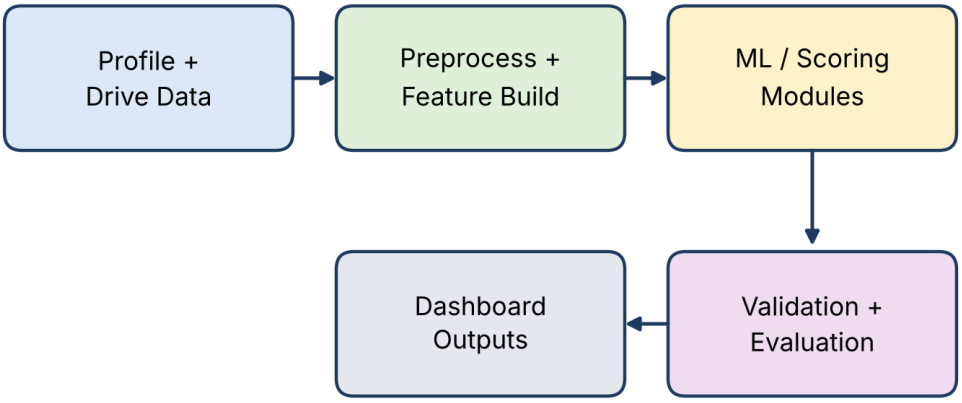


Figure 3 Shared Data-To-Insight Workflow for The Proposed ML Modules

2.5.Company-Drive Priority Classification

For a student *i* and drive *j*, the system builds a relevance score from domain similarity, required-skill overlap and role relevance. A transparent starting score is $R(i,j) = w_1D(i,j) + w_2S(i,j) + w_3J(i,j)$, where *D* is domain relevance, *S* is the proportion of required skills the student has demonstrated, *J* is role/profile relevance, and $w_1+w_2+w_3=1$. The weights have to be chosen using expert review and validation data. The score is mapped to High, Medium or Low with thresholds set during validation. The interface can display the matched skills and the drive skills still missing, so the priority label can be explained. When labelled student–drive preference data are not available, a rule-based weighted score is a suitable baseline. Once enough reviewed labels or student-declared preferences have been collected, Logistic Regression, Decision Tree, Random Forest or SVM can be evaluated as classification alternatives. A priority label must never be described as the probability that a company will select the student.

2.6.Domain Progress and Skill Gap Analysis

Each supported domain is represented by a curated list of skills, each with an agreed importance level. Let *N* be the number of skills in the selected domain and *C* the number of skills supported by evidence. A basic progress measure is Progress(%) = $(C/N) \times 100$, and the remaining percentage is 100 − Progress(%). Evidence can include practical assessments, completed projects, certifications or verified learning records. A skill that is only entered as profile text should not automatically count as proven proficiency. The output lists completed skills, skills to improve and remaining skills, ordered by their relevance to active company drives.

2.7.Resume Strength Analysis

Once a student uploads a resume, a document extraction stage pulls the text out of supported PDF or DOCX files. The text is then parsed to find education, technical skills, projects, internships or experience, certifications and domain-related terms. A transparent rubric can give category-wise feedback on technical-skill relevance, project evidence, experience, certifications and resume completeness. The overall score is a weighted rubric score and not a measure of guaranteed employability. The report should explain why each category received its score and offer specific suggestions for improvement. Parsing errors and unsupported layouts must be reported to the user rather than silently interpreted.

2.8.Placement Preparation Recommendation

Preparation topics are ranked by comparing the student's profile with the requirements of a selected role. A content-based baseline can use TF–IDF vectors and cosine similarity. For topic *t*, job role *j* and student *i*, $s(t|i,j) = \lambda \cos(v_t,v_j) + (1-\lambda) \cos(v_t,v_i)$, where v_t represents a topic, v_j the role requirements and v_i the student's skills, projects and domain interests. The top-*K* topics are returned. The list can also take in the missing skills found by the domain progress module and the requirements extracted from active drives.

2.9.Placement Outcome Prediction and Analytics

Where enough labelled historical records exist, classification models can estimate a placed/not-

placed outcome using only information available before the outcome. Candidate features are CGPA, school marks, standing arrears, skills, domain interests, project count, certification count, experience, and department or batch. Company, package, offer-letter and other post-outcome fields must be kept out of the predictors to prevent label leakage. If enough placed records are available, package estimation can be explored as a separate regression task. Descriptive placement rates and package aggregates stay available whether or not predictive modelling is used. For a group g, the placement rate is $PR_g = (N_{placed,g} / N_{total,g}) \times 100$. When only a few yearly observations exist, trend analysis should use simple moving averages or linear trend fitting. More complex forecasting makes sense only when the amount and quality of the time-series data justify it.

2.10. Technology Stack and Data Flow

Table 2 Technology Stack and its Purpose in HireMind

Technology	Purpose in HireMind
HTML, CSS and existing web pages	Render student, staff and administrator screens; React can support richer interactive components.
Node.js and Express	Handle authentication, role-based routes, business rules and API coordination.
MySQL	Store student profiles, skills, domains, drives, requirements, resumes metadata and placement outcomes.
Python, Pandas and NumPy	Prepare structured data, clean records and construct model features.
Scikit-learn	Implement baseline ranking/classification/regression models and evaluate them.
FastAPI	Expose ML functions as internal APIs called by the Node.js backend.
PDF/DOCX extraction libraries	Extract resume text before structured analysis.

The expected request flow is as follows. The user updates a profile or opens a drive, and the Node.js backend retrieves the permitted records from MySQL. The backend sends only the feature data that is needed to the FastAPI service, the ML service computes a result, and the backend returns that result to the web interface. Sensitive resume content should be processed with access control, limited retention and explicit institutional approval. MySQL stays the primary application database, and the initial design does not need a separate ML database.

2.11. Data and Labels

The existing schema offers candidate attributes such as CGPA, 10th and 12th marks, standing arrears, skills text, domain interests, project, certification and experience counts, department and batch. On the drive side, the attributes are role, job_skills, requirements, eligibility text, package, location and eligible departments. Some tasks need additional labels: human-reviewed relevance judgements for recommendation ranking, student-declared or expert-reviewed preference labels for High/Medium/Low classification, and evidence-based skill proficiency labels for progress analysis. Resume scoring needs a documented rubric that humans have reviewed. A public campus-placement dataset may serve as an external benchmark for outcome classification, but its downloaded file, field definitions, class distribution and licensing must be checked before use, and it must not be presented as institutional data. Any college records should be anonymized and used only with the required approvals.

2.12. Validation and Metrics

For classification, we report accuracy, per-class precision, recall and F1, macro-averaged scores and a confusion matrix. Company-priority ranking is assessed with Precision@K, Recall@K and NDCG@K against reviewed relevance labels, and package regression with MAE and RMSE. For domain progress and resume analysis, automated assessments are compared with independent human or practical-assessment judgements, and we report

agreement and category-level error. A temporal split by graduating batch is used where possible; otherwise we use stratified train/test splitting and stratified cross-validation. Resampling and hyperparameter tuning must happen only inside the training folds.

Each model should be compared with a simple baseline, such as majority-class prediction or keyword overlap. Fairness checks should look at error differences across groups wherever this is lawful, appropriate and sufficiently supported by data. Predicted labels should be advisory and come with explanations, and they should not be the only basis for placement eligibility or student evaluation.

3. Results and Discussion

3.1.Results

The supplied prototype description verifies the management foundation and the descriptive placement aggregates. The ML modules in this paper are proposed designs, and in the supplied materials no model has been trained or evaluated on HireMind data. For this reason the paper reports no accuracy, F1, MAE, RMSE, ranking scores or validated resume-strength score. Any sample percentages in interface mockups are illustrative only.

3.2.Discussion

The main limitations are that the inspected code has no implemented ML endpoints, that preference and skill-evidence labels still have to be collected, that resume parsing may make errors, that institutional sample sizes are limited, that historical placement records may carry bias, and that student data raises privacy requirements. Company priority is a relevance aid and must not be read as a recruitment guarantee. Domain progress depends on the quality of the skill catalogue and the proficiency evidence, and resume scores depend on how transparent and well validated the scoring rubric is.

Conclusion and Future Work

This paper presents an updated HireMind design that links placement management with machine-learning-supported guidance for students. The proposed extension adds company-drive prioritization, domain skill progress and gap analysis, resume strength feedback, preparation recommendations and placement outcome analytics. It keeps the existing Node.js, Express and MySQL foundation and proposes Python-based ML services behind FastAPI. Future work includes finalizing the domain skill catalogue, collecting reviewed labels, implementing the ML APIs, integrating the student and staff interfaces, running leakage-aware and fairness-aware evaluation, and reporting measured results. We will make performance claims only after experimental validation.

References

[1]. X. Zhong, “University student management system based on deep learning algorithm,” in Proc. 2024 Int. Conf. Interactive Intelligent Systems and Techniques (IIST), IEEE, 2024, pp. 616–620, doi: 10.1109/IIST62526.2024.00065.

[2]. K. D. Thilak, K. L. Devi, K. Kalaiselvi, and J. Teja, “Revolutionizing university graduate employability: Leveraging advanced machine learning models to optimize campus recruitment and placement strategies,” in Proc. 2023 Int. Conf. Research Methodologies in Knowledge Management, Artificial Intelligence and Telecommunication Engineering (RMKMATE), IEEE, 2023, doi: 10.1109/RMKMATE59243.2023.10369300.

[3]. V. Yadav, S. R. Rauniyar, U. Gewali, A. Shakya, and S. Khatri, “Smart job recruitment automation: Bridging industry and university,” in Proc. 2019 Artificial Intelligence for Transforming Business and Society (AITB), IEEE, 2019, pp. 39–43, doi: 10.1109/AITB48515.2019.8947445.

[4]. A. Sunil, V. S. ATS, and V. Chintham, “Adaptive fuzzy campus placement based optimization algorithm,” in Proc. 2023 5th Int. Conf. Energy, Power and Environment: Towards Flexible Green Energy Technologies (ICEPE), IEEE, 2023, doi: 10.1109/ICEPE57949.2023.10201630.