WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

NIPUN AI: One Platform Every Placement Needs An AI-Powered Unified Placement Readiness System

Mr. Akhilesh Sathyanarayan [a], Shree Padmavathi S [b], Darshan Bharadwaj [c], Sameera B B [d]

[a] Guide Training and Placement Officer & Assistant Professor, Dept. of CSE, Jyothy Institute of Technology, Bangalore, India
[b,c,d] Dept. of CSE, Jyothy Institute of Technology, Bangalore, India
Email: placement@jyothyit.ac.in, 1jt22cs125@jyothyit.ac.in, 1jt22cs101@jyothyit.ac.in, 1jt22cs132@jyothyit.ac.in

A B S T R A C T

Across many campuses, the journey from learning to earning is still stitched together from disconnected tools for resume writing, coding practice, job search, and interview preparation. This fragmentation leads to confusion, duplicated effort, and under-prepared graduates entering a highly competitive job market. NIPUN AI (Next-Gen Intelligent Platform for Upskilling and Placement Navigation using Artificial Intelligence) is designed as a unified, AI-powered placement-readiness platform that consolidates these activities into a single ecosystem. The system combines four core capabilities: an ATS-driven resume optimizer that scores and refines student resumes by domain; a one-click job-application workflow that connects students, placement cells, and companies; a coding practice module with curated DSA problems and live execution; and an AI mock interview engine that evaluates HR and technical responses using local large language models and behavioural cues. Built on a modular React–Node–MongoDB stack with microservices for resume parsing, job matching, coding evaluation, and interview feedback, NIPUN AI offers role-based dashboards for students and administrators, continuous progress tracking, and data-backed insights on placement readiness. This paper presents the design, architecture, and implementation of NIPUN AI, outlines its evaluation strategy using composite readiness scores and usage analytics, and situates the platform in the context of existing placement management and recommendation systems.

Keywords: Campus placements, Artificial Intelligence, Resume parsing, ATS scoring, Mock interviews, Recommendation systems, DSA practice, Web application.

1. Introduction

Campus placements are often the first serious bridge between academic training and industry expectations, yet the infrastructure that supports this transition is usually fragmented. Students jump between learning platforms, resume builders, coding sites, job portals, and ad-hoc mock interview tools. None of these systems share context, which means that feedback from one stage rarely informs the next. Placement officers, on the other hand, rely on spreadsheets or legacy portals that provide limited analytics and almost no personalised mentoring signals.

NIPUN AI (Next-Gen Intelligent Platform for Upskilling and Placement Navigation using Artificial

Intelligence) is conceived as a single, intelligent interface that centralises the placement journey. The word “Nipun” means “skilled” in Sanskrit and reflects the platform’s primary goal: to help students move from classroom knowledge to industry-ready skills in a structured, measurable way. At its core, NIPUN AI combines four pillars:

- a domain-aware ATS resume scanner that analyses structure, skills, and keyword alignment;
- a one-click application pipeline where students apply once and the admin orchestrates forwarding to companies;
- a coding practice module with 50 curated DSA questions and live execution through an online judge;
- an AI mock interview engine that simulates HR and technical interviews and generates feedback on clarity, content, and confidence.

By bringing these features under one roof and exposing them through role-based dashboards, NIPUN AI aims to reduce friction for students, offer richer signals to placement cells, and create an iterative loop of training, testing, feedback, and application rather than one-off events.

2. Literature Survey

Digital placement portals and AI-based recommendation engines have been studied extensively in recent years. However, most systems either focus on workflow automation for placement cells or on narrow tasks such as job recommendation or prediction of placement outcomes. We review key contributions that inform the design of NIPUN AI.

1) Enhancing Student Placement with Cross-Platform Application

Authors: M. D. Bhalerao, R. S. Londhe, R. B. Yadav.

Venue: IJCRT, Vol. 12, Issue 2, 2024 [1].

Brief: This work presents a cross-platform web and mobile application that improves accessibility for students and recruiters. Students can update profiles, browse openings, and apply online, while placement officers manage drives and track participation. The focus is on integrated access rather than deep AI personalisation.

2) College Placement Management System

Authors: Keerthana M. M. et al.

Venue: IJCRT, Feb. 2024 [2].

Brief: The authors design a web-based dashboard that digitises core placement cell operations, including student registration, resume upload, job notifications, and recruiter coordination. The system effectively replaces manual paperwork but does not provide analytics-driven upskilling guidance.

3) Maximizing Campus Placement Through Machine Learning

Authors: Sarita Byagar et al.

Venue: Journal of Advanced Zoology, Vol. 45(S-4), 2024 [3].

Brief: This paper explores machine-learning models for predicting placement success and recommending training plans. It highlights the potential of data-driven insights for placement preparation, but remains largely experimental and detached from a full-stack portal.

4) College Training and Placement System

Authors: Anonymous.

Venue: Preprint, Nov. 2024 [4].

Brief: The system uses cloud-based tools to manage placement workflows, emphasising real-time updates and improved accessibility for stakeholders. It primarily addresses process automation instead of personalised student mentoring.

5) Placement-Cell Management System

Authors: Nitin Pathak et al.

Venue: IRJMETS, May 2024 [5].

Brief: This web-based platform monitors student progress, recruiter interaction, and placement drives through dashboards. It improves transparency and monitoring but does not integrate coding practice or interview simulation.

6) Smart Campus 4.0: Digitalization with Industry 4.0

Authors: Anonymous.

Venue: ARASET, 2023 [6].

Brief: The work discusses how IoT, AI, and cloud computing can be combined to build Smart Campus ecosystems. Placement services are treated as one of several modules, signalling the importance of scalable infrastructure but not a dedicated AI mentor.

7) Placement Automation System for Educational Institutes

Authors: Anonymous.

Venue: IJCRT, May 2023 [7].

Brief: This portal automates interview scheduling, shortlisting, and recruiter notifications. The centralised workflow reduces manual work for administrators but relies mostly on rule-based logic.

8) Training and Placement Cell Automation at AMC Engineering College

Authors: Amit Kumar Singh et al.

Venue: Project report, May 2023 [8].

Brief: The system stores all training and placement data digitally with role-based access for students and TPOs, emphasising secure operations and communication.

9) A Comprehensive Campus Recruitment and Placement System

Authors: Anonymous.

Venue: IJRASET, Apr. 2023 [9].

Brief: A modular platform that offers student registration, job posting, application tracking, and recruiter feedback. The design improves transparency and reporting but lacks advanced AI-driven scoring and feedback.

10) AI-Based Smart Placement Recommendation System

Authors: R. Patel, A. Shah, M. Sharma.

Venue: IJERT, Vol. 11(5), 2022 [10].

Brief: This paper introduces AI-based recommendation techniques to match candidates with suitable job opportunities. It takes a step towards intelligent guidance, but focuses mainly on job recommendation rather than unifying resume analysis, coding practice, and interview feedback.

Across these works, we observe three limitations: most systems digitise workflows without deep intelligence, analytics components are often isolated from user-facing portals, and very few solutions provide continuous, student-centric mentoring. NIPUN AI is positioned to bridge these gaps through an integrated ATS, coding, and AI interview ecosystem.

2. Objectives

(O1) Unified placement-readiness platform: Design a single portal that consolidates resume optimisation, job applications, coding practice, and AI mock interviews for students, with a complementary admin dashboard for placement officers.

(O2) Domain-aware ATS resume scanning: Develop an intelligent ATS engine that analyses uploaded resumes with respect to selected domains (e.g., AI/ML, Web Development, Data Science, Cyber Security), producing a score, keyword match report, and concrete improvement suggestions.

(O3) One-click job application workflow: Implement a job portal where students can apply to posted roles in one click, while admins view consolidated applicant lists, filter candidates, and forward shortlisted profiles to recruiters.

(O4) Structured coding practice module: Provide at least 50 curated DSA problems with integrated code execution, submission tracking, and basic analytics to support regular problem-solving practice. **(O5) AI-driven mock interview and feedback:** Build an AI mock interview system that evaluates HR and technical responses using local LLMs (e.g., Ollama) and behavioural signals, returning detailed feedback on clarity, correctness, confidence, and communication.

(O6) Data-backed progress tracking: Expose composite readiness scores, history of attempts, and module-wise analytics to students and placement officers through dashboards, enabling continuous monitoring and targeted intervention.

3. Proposed Methodology

The proposed methodology follows a modular client–server design where independently deployable services collaborate via a REST-based API gateway. The platform is split into student and admin modules, each exposing focused workflows while sharing a common data and AI backbone.

A. ATS Resume Scanner

Students upload resumes in PDF or DOC format. A Python-based parser extracts education, skills, projects, and experience, normalises the text, and compares it with domain-specific keyword templates and role descriptions. The ATS engine computes:

- an overall ATS score;
- domain keyword coverage and missing keywords;
- structural checks (section ordering, contact information, action verbs).

The output is stored as structured JSON and surfaced as a detailed feedback report on the dashboard.

B. One-Click Job Application System

Admins create and manage job postings with role descriptions, eligibility criteria, and required skills. Students view a personalised feed of openings filtered by branch, graduation year, and domain interests. When a student clicks “Apply”, the system links the user’s profile, current resume, and ATS data with the job entry, logging an application that the admin can review, shortlist, and export.

C. Coding Practice and Evaluation

A curated set of 50 foundational DSA questions is organised by topic and mapped to difficulty levels. The coding interface connects to an execution engine (such as Piston API) to compile and run submitted code in languages like C++, Java, or Python. For each problem, the system stores:

- number of attempts and success status;
- execution results and runtime error logs;
- topic-wise progress indicators.

This data feeds into the readiness index described later.

D. AI Mock Interview Engine

The AI mock interview subsystem supports HR and basic technical interviews. Question sets are chosen according to the student’s domain and past performance. User responses are captured as text (or speech-to-text in extended versions) and passed to a locally hosted LLM via Ollama. The model generates feedback along dimensions such as content correctness, communication clarity, and confidence markers. Scores and qualitative comments are saved per session.

E. Admin Analytics and Dashboards

Placement officers access an admin dashboard that aggregates:

- ATS statistics (average scores, common gaps);
- coding practice metrics;
- interview performance distributions;
- job application counts and conversion rates.

These insights help schedule focused training sessions and track batch progress over time.

4. Implementation

NIPUN AI is implemented using a MERN-inspired stack with Python-based auxiliary services. The design emphasises modularity, reproducibility, and compatibility with campus-scale deployments.

A. Development Environment and Frameworks

The core components are built in JavaScript and Python:

- **Frontend:** React.js and Tailwind CSS for responsive, component-based UI.
- **Backend:** Node.js with Express.js to expose REST APIs and act as API gateway.
- **Database:** MongoDB Atlas for scalable, document-oriented storage.
- **AI/ML:** Python microservices using NLP libraries and Ollama-backed LLMs for feedback genera- tion.
- **Coding Engine:** Integration with an execution API (e.g., Piston) for live code runs.

Authentication is implemented through JWTs, and environment-specific secrets are handled via configuration files or environment variables.

B. Data Flow and System Pipeline

When a student logs in, the frontend requests profile and dashboard data via the API gateway. For ATS scanning, the resume upload endpoint forwards the file to the Python parser, which returns structured fields and scores that are persisted to MongoDB. Coding submissions are sent to the execution API and the results are recorded as attempt logs. Mock interview responses are sent to the AI evaluation microservice, which stores scores and feedback. Admin actions such as posting jobs or viewing analytics use the same gateway but operate on different collections and views.

C. ATS and Job-Matching Service

The resume microservice tokenises and normalises resume content, then matches it against domain-specific keyword sets and job templates. Each job application records both raw ATS metrics and a job-specific match score, allowing admins to sort applicants beyond CGPA or branch filters.

D. Coding and Interview Modules

The coding module maintains problem metadata, language options, and expected outputs. The interview module stores question banks and uses LLM prompts tailored to placement interviews (e.g., behavioural questions, basic CS fundamentals). Each completed session generates an entry with a numeric score and narrative feedback.

E. Temporal Logs and Progress Tracking

All activities—scans, submissions, interviews, and applications—are timestamped. Dashboards aggregate this history into visualisations such as progress charts and readiness indicators, helping students plan their preparation schedule.

F. Inference Strategy

Although NIPUN AI does not perform heavy GPU-based diffusion, the platform still requires consistent behaviour from its AI components. For mock interviews, prompts are constructed in a templated fashion:

Prompt = f (Question, UserAnswer, Rubric, DomainTags), (1) where the rubric encodes evaluation criteria such as technical depth, structure, and communication.

This design keeps the evaluation deterministic for a given model and version.

G. Evaluation Metrics

To summarise a student's placement readiness, we define an aggregate score that combines ATS perfor- mance, coding results, and interview feedback. Let S_r be the normalised resume score, S_c the coding score (e.g., ratio of solved problems), and S_i the interview score. A weighted readiness index is given by:

$$\mathrm{RI} = w_r S_r + w_c S_c + w_i S_i, \quad (2)$$

with $w_r + w_c + w_i = 1$. Typical settings might assign higher weight to coding and interview performance in later stages of preparation. Additional metrics include application-to-shortlist ratios and time-series plots of RI to capture improvement trends.

H. User Interface and Experience

The UI is organised into a three-panel layout:

- a navigation sidebar for modules (Dashboard, Resume, Coding, Interviews, Jobs);
- a main content area displaying cards and tables;
- a compact header with notifications and profile options.

Students can quickly jump from feedback to corresponding actions (e.g., from a resume gap to editing the resume, or from a weak topic to a coding problem list).

I. Reproducibility and Experiment Tracking

Configuration of models, weightings, and question sets is externalised in JSON files so that different experimental setups (e.g., alternative scoring rubrics) can be replayed. Logs of AI prompts and responses are stored with anonymised identifiers for debugging and improvement while preserving privacy.

J. Performance and Hardware Requirements

The platform is designed to run comfortably on mid-range servers. A typical deployment uses an 8–16 GB RAM machine for the Node.js and MongoDB components and a separate GPU-enabled host or local workstation for Ollama-based inference. For college-scale usage, horizontal scaling of stateless services and managed database tiers can be adopted.

5. System Architecture

Overview

NIPUN AI follows a modular client–server architecture where the React frontend communicates with an Express.js API gateway. The gateway routes authenticated requests to backend services responsible for ATS analysis, job management, coding practice, and interview evaluation. Figure 1 illustrates the high-level architecture.

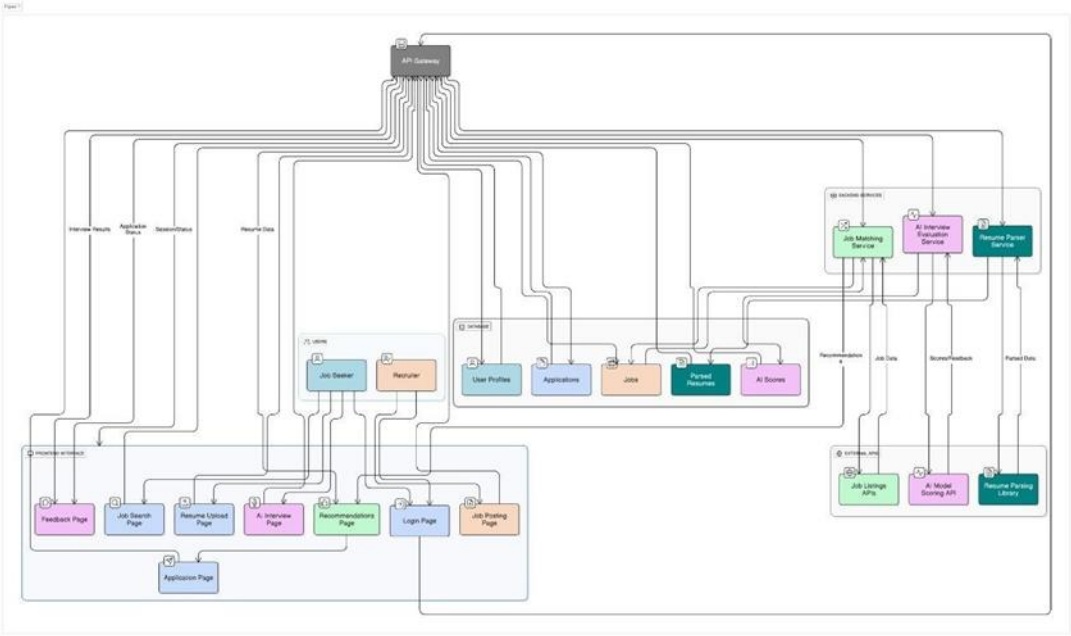

Figure 1: NIPUN AI high-level architecture: student/admin clients connect to an API gateway, which orchestrates ATS scanning, job services, coding engine, and AI interview evaluator over a shared MongoDB datastore.

Student and Admin Clients

Students and admins access the system via web browsers. Students interact with modules for profile setup, ATS scans, coding, mock interviews, and job applications. Admins manage postings, upload resources, and monitor batch progress. Both roles rely on JWT-secured sessions.

Core Backend Services

The backend is decomposed into logical services:

- **Resume Service:** handles file uploads, invokes the parser, and stores structured resume data.
- **Job Service:** stores listings, eligibility criteria, and application records.
- **Coding Service:** interfaces with the execution API and tracks submissions.
- **Interview Service:** coordinates question selection and LLM-based feedback.
- **Admin Service:** aggregates analytics and exposes dashboard queries.

These services share MongoDB collections but can be factored into separate microservices as the system scales.

Data Storage Layer

MongoDB Atlas holds collections for users, resumes, jobs, applications, coding attempts, and interview sessions. Indexes on user ID, job ID, and timestamps enable efficient queries for dashboard views and batch analytics.

6. Conclusion

NIPUN AI demonstrates how AI-driven automation and thoughtful system design can unify fragmented placement activities into a single, student-centric ecosystem. By combining ATS-based resume evaluation, structured coding practice, AI-led mock interviews, and streamlined job applications, the platform supports a continuous loop of preparation, feedback, and opportunity discovery. For institutions, role- based dashboards and analytics provide a clearer view of batch readiness and training needs.

Future work includes tighter integration of behavioural signals (such as webcam-based gaze tracking), richer company-specific assessments, multilingual support, and deeper recommendation models that align learning resources with individual gaps. As such features are added, NIPUN AI has the potential to function not just as a placement portal, but as a long-term digital mentor guiding students from their first year of study to their first offer letter.

References

[1] M. D. Bhalerao, R. S. Londhe, and R. B. Yadav, “Enhancing Student Placement with Cross-Platform Application,” *IJCRT*, vol. 12, no. 2, 2024.

[2] M. M. Keerthana et al., “College Placement Management System,” *IJCRT*, Feb. 2024.

[3] S. Byagar et al., “Maximizing Campus Placement Through Machine Learning,” *Journal of Advanced Zoology*, vol. 45(S-4), 2024.

[4] Anonymous, “College Training and Placement System,” Preprint, Nov. 2024.

[5] N. Pathak et al., “Placement-Cell Management System,” *IRJMETS*, May 2024.

[6] Anonymous, “Smart Campus 4.0: Digitalization of University Campus with Assimilation of Industry 4.0,” *ARASET*, vol. 32, no. 1, 2023.

[7] Anonymous, “Placement Automation System for Educational Institutes,” *IJCRT*, May 2023.

[8] A. K. Singh et al., “Training and Placement Cell Automation at AMC Engineering College,” Project report, May 2023.

[9] Anonymous, “A Comprehensive Campus Recruitment and Placement System,” *IJRASET*, Apr. 2023.

[10] R. Patel, A. Shah, and M. Sharma, “AI-Based Smart Placement Recommendation System,” *IJERT*, vol. 11, no. 5, pp. 155–160, 2022.