IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 4, July 2026
Impact Factor: 8.2

Placement Prediction and Management System: A Role-Based Web Platform for Automating Campus Recruitment Workflows

Syed Farhan[1], Varshitha MR[2], Sinchana TM[3], Poornimaha HA[4]
[*1,2,3]Student, Department of Computer Applications
[*4]Assistant Professor, Department of Computer Applications
Maharaja Institute of Technology Mysore, Mandya, Karnataka, India.

Abstract: *Campus placement remains one of the most administratively demanding activities undertaken by academic institutions, requiring coordinators to reconcile student eligibility, recruiter requirements, and scheduling constraints, typically through a patchwork of spreadsheets, notice boards, and email threads. This fragmentation obscures a student's real-time placement readiness and leaves recruiters and coordinators without a consolidated view of the recruitment pipeline. This paper presents the Placement Prediction and Management System, a role-based web application that consolidates student profile management, recruiter job postings, coordinator oversight, and administrative control within a single platform. The system is built on a Node.js and Express.js backend, renders views through the EJS templating engine, and persists data in MongoDB. Authentication is implemented through server-side sessions combined with cookie-based state and one-time-password verification delivered by email, avoiding reliance on third-party identity providers. A distinguishing component of the platform is a rule-based placement prediction engine that computes a readiness score for each student from weighted academic and extracurricular indicators—cumulative grade point average, active backlog count, certification density, and prior application history—surfacing this score to students, recruiters, and coordinators before a placement drive is scheduled. Resume and certificate uploads are handled through an in-application file-upload middleware, and platform-wide notifications keep students informed of new postings, shortlist outcomes, and drive schedules. The system was implemented and functionally validated across its four user roles—student, recruiter, coordinator, and administrator—with attention to session integrity, upload handling, and score computation correctness. This paper documents the architectural rationale, implementation details, functional evaluation, and limitations of the platform, and outlines a path toward integrating a learned prediction model in place of the current rule-based heuristic.*

Keywords: Placement management system, placement prediction, campus recruitment, MERN-adjacent stack, Node.js, Express.js, MongoDB, role-based access control, OTP authentication, rule-based scoring

1. INTRODUCTION

Placement drives sit at the intersection of institutional reputation and individual career outcomes, yet the process by which they are administered has changed little at many institutions: eligibility lists are compiled manually from mark sheets, job postings circulate through notice boards or messaging groups, and a student's likelihood of being shortlisted is rarely made transparent until a recruiter's criteria are applied after the fact. This opacity is compounded by scale

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 4, July 2026 Impact Factor: 8.2

— a single training-and-placement cell may coordinate several hundred students against dozens of recruiters within a single academic term, and manual reconciliation of eligibility, application status, and drive scheduling does not scale gracefully against that volume.

A further consequence of this fragmentation is that students receive little actionable signal about their own placement readiness until they are already being evaluated by a recruiter. A student with a borderline number of backlogs, or a thin certification record, has no accessible way to gauge how these factors weigh against a specific opportunity, and a coordinator has no consolidated view that would let them flag such gaps early enough for intervention.

This paper describes a web-based Placement Prediction and Management System that addresses both the coordination and the transparency problem within a single platform. The system separates concerns across four user roles — student, recruiter, coordinator, and administrator — each operating through a dedicated dashboard, while a shared backend enforces consistent business rules across all four. Rather than treating placement readiness as an implicit judgment made only at the recruiter's end, the platform computes an explicit readiness score from a small set of weighted, auditable factors and surfaces it before a drive is scheduled, giving students early visibility and giving coordinators a basis for shortlisting that does not depend on manually cross-referencing academic records against recruiter criteria.

The remainder of this paper is organised as follows. Section
II surveys related work on placement and recruitment automation. Section III states the problem this project addresses. Section IV outlines the objectives and scope of the system. Section V details the system architecture and methodology. Section VI describes the implementation. Section
VII reports functional evaluation results, and Section VIII concludes with limitations and future directions.

II. RELATED WORK

Research and applied work relevant to this project falls broadly into three groups: institutional placement-management portals, general e-recruitment platforms, and predictive approaches to candidate–role matching.

A number of college placement portals have been documented in the literature as digitising the coordinator's administrative workload — centralising student records, job notifications, and drive scheduling in place of notice boards and spreadsheets. These systems typically emphasise record-keeping and workflow digitisation, with role-based access separating student-facing and coordinator-facing views, but placement outcomes in these designs are generally recorded retrospectively rather than estimated in advance of a drive.

A second body of work addresses e-recruitment more generally, focusing on matching a candidate's résumé or profile against a job description through keyword or similarity-based techniques. These approaches are effective at ranking applicants against a specific posting but are not typically framed around a student's overall, drive-independent placement readiness, and are less commonly integrated with the day-to-day administrative functions — OTP-secured registration, certificate verification, and coordinator dashboards — that an institutional placement cell requires.

A third strand considers predictive modelling of placement or hiring outcomes, most commonly through supervised learning over historical placement records to estimate a student's probability of being placed or a candidate's suitability for a role. Such approaches can achieve strong predictive accuracy where sufficient historical, labelled data exists, but they introduce a dependency on a data-science pipeline separate from the core web application, and their outputs can be difficult for a non-technical coordinator to audit or explain.

The present work is positioned between these strands: it adopts the administrative completeness of institutional placement portals — registration, profile management, recruiter and coordinator workflows — while incorporating an explicit, criterion-based readiness score in place of a purely retrospective record. Rather than a learned model requiring a labelled historical dataset that a newly deployed system would not yet possess, the platform uses a transparent, rule-based scoring formula whose weights are visible and adjustable by an administrator, trading some predictive sophistication for interpretability and immediate deployability — a trade-off the paper returns to in Section VIII.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 4, July 2026
Impact Factor: 8.2

III. PROBLEM STATEMENT

Institutional placement coordination currently depends on manually reconciling three separate concerns — student eligibility and profile data, recruiter requirements and job postings, and drive scheduling and outcome tracking — usually across disconnected tools such as spreadsheets, email, and physical notice boards. This fragmentation produces two concrete problems. First, coordinators lack a single, authoritative view of which students are eligible for which postings at any given time, making shortlisting a manual and error-prone exercise as student numbers grow. Second, students have no transparent, ongoing indication of their own placement readiness, learning of eligibility gaps — insufficient CGPA, unresolved backlogs, missing certifications — only when it is too late to address them before a specific drive. A unified, role-aware platform that consolidates these workflows and computes an explicit, auditable readiness score for each student would address both problems while remaining deployable without a pre-existing historical placement dataset.

Each layer communicates with its neighbours through well-defined interfaces so that, for example, the scoring formula used in the prediction module can be revised without altering the presentation templates or the storage schema.

IV. OBJECTIVES AND SCOPE

A. Objectives

• Develop a full-stack, role-based web application serving student, recruiter, coordinator, and administrator users through distinct dashboards on a shared backend.

• Implement secure registration and login using session-based authentication, cookie-managed state, and email-delivered one-time-password verification.

• Provide student-side profile management, including résumé and certificate uploads, that recruiters and coordinators can review during shortlisting.

• Allow recruiters to publish job postings with eligibility criteria, and allow coordinators to review, schedule, and track placement drives against those postings.

• Compute a transparent, weighted placement-readiness score per student from academic and profile indicators, and surface it before a drive is scheduled.

• Deliver in-application and email notifications for new postings, shortlist outcomes, and drive schedules, supported by a lightweight messaging module.

• Maintain administrator-level oversight of users, postings, and platform statistics.

B. Scope

The platform targets a single institution's training-and-placement cell as its primary deployment context, with four user roles operating over a shared MongoDB data store. The placement-prediction component in the present implementation is a rule-based scoring engine rather than a trained machine-learning model, reflecting the absence of a sufficiently large historical, labelled dataset at initial deployment; Section VIII discusses the conditions under which a learned model could subsequently replace or augment this engine. The system is scoped to text-and-document based workflows — profile data, uploaded files, and structured job postings — and does not currently address video interviewing or third-party job-portal integration.

V. SYSTEM ARCHITECTURE AND METHODOLOGY

The platform follows a layered architecture comprising a presentation layer, an application layer, a prediction and notification layer, and a storage layer, illustrated in Figure 1.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 4, July 2026
Impact Factor: 8.2

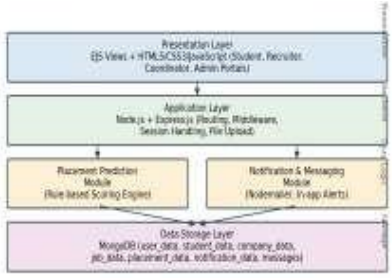

Figure 1. Four-layer system architecture: presentation, application, prediction/notification, and storage.

A. Presentation Layer

The client-facing views are rendered server-side using the EJS templating engine, with HTML5, CSS3, and vanilla JavaScript (ES6) providing structure, styling, and interactivity respectively. Rendering views on the server keeps the client lightweight and avoids the build tooling associated with a single-page-application framework, which was judged unnecessary for a role-based, form-and-dashboard-centric application of this kind.

B. Authentication and Session Management

User registration begins with email-based one-time-password verification, implemented through Nodemailer, before an account is activated. Once verified, authentication state is maintained through express-session on the server and a corresponding cookie on the client, managed via cookie-parser. Passwords are stored using Base64 encoding of credential buffers as an interim obfuscation measure; this design choice and its limitations are discussed further in Section VIII. Delegating identity verification to an OTP step, rather than to a third-party identity provider, kept the authentication flow self-contained within the Node.js/Express stack and avoided an external service dependency.

C. Backend and Application Logic

Express.js structures the application around a router-per-role convention, with separate route modules for student, recruiter, coordinator, and administrator actions, each guarded by session-aware middleware that restricts access according to the authenticated user's role. Request bodies are parsed via express.urlencoded and express.json, and file payloads — résumés and certificates — are handled through express-fileupload middleware, which streams uploaded files to a designated storage path on the server for later retrieval.

D. Placement Prediction Engine

The prediction module computes a per-student readiness score against a job posting from a small set of weighted indicators: cumulative grade point average, active backlog count, certification and skill-record density, and prior application and shortlist history. Each indicator is normalised to a common scale and combined through a fixed set of weights configurable by an administrator, producing a single readiness figure that is displayed to the student and made visible to the coordinator during shortlisting. Because the formula and its weights are stored and computed directly within the application rather than inferred by an opaque model, an administrator or coordinator can inspect exactly why a given score was produced, which was judged important for a context where a student's practical outcomes are affected by the score.

E. Data Storage

MongoDB serves as the system's persistence layer, accessed through the native MongoDB Node.js driver rather than an object-document-mapping library, with CRUD operations expressed directly through findOne, insertOne, updateOne, and deleteOne calls against a defined set of collections — user_data, student_data, company_data, job_data, placement_data, notification_data, and messages. This direct-driver approach was chosen to keep the data-access layer transparent and to avoid the additional abstraction and schema-migration overhead of an ODM during early, fast-iterating development.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 4, July 2026
Impact Factor: 8.2

F. Notification and Messaging Subsystem

New job postings, shortlist decisions, and drive schedules trigger both an in-application notification record and, where configured, an outbound email via Nodemailer. A lightweight messaging module additionally allows coordinators and recruiters to exchange short messages with students regarding drive logistics, persisted in the messages collection.

G. End-to-End Flow

The complete sequence, illustrated in Figure 2, proceeds as follows: (1) registration and OTP verification; (2) session-based login; (3) profile creation, including résumé and certificate upload; (4) a recruiter publishing a job posting; (5) the prediction engine computing a readiness score for each eligible student; (6) the coordinator reviewing the resulting shortlist and scheduling a drive; (7) students applying to or being notified of the posting; and (8) the recorded placement outcome feeding back into the dashboard and platform statistics.

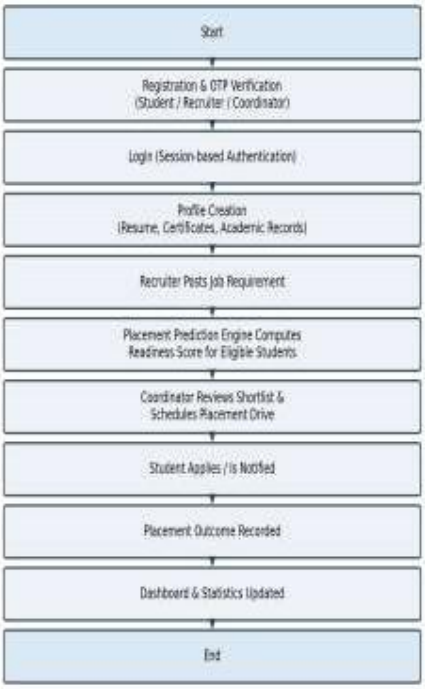

Figure 2. End-to-end sequence from registration through placement-outcome recording.

VI. IMPLEMENTATION DETAILS

A. Technology Stack

- Frontend: HTML5, CSS3, JavaScript (ES6), EJS
- Backend: Node.js, Express.js
- Database: MongoDB, accessed via the native mongodb Node.js driver
- Authentication: express-session, cookie-parser, Base64-encoded credentials, Nodemailer-based OTP
- File Handling: express-fileupload, Node's fs and path modules
- Deployment/Runtime: Localhost Express server over Node.js's HTTP module
- Tooling: Visual Studio Code, MongoDB Compass, npm, Git/GitHub, Google Chrome and Chrome DevTools

B. Module-Wise Description

The Student module covers registration, OTP verification, profile management, résumé and certificate upload, viewing job postings and readiness scores, and receiving notifications. The Recruiter module covers job posting, eligibility-criteria specification, and reviewing scored, shortlisted candidates. The Coordinator module covers cross-cutting

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 4, July 2026 Impact Factor: 8.2

oversight of eligible students, drive scheduling, gallery and statistics views, and messaging with both students and recruiters. The Administrator module covers user and role management, platform-wide statistics, and configuration of the prediction engine's scoring weights.

C. Database Design

Each of the seven MongoDB collections is scoped to a single concern: user_data holds shared account and role information; student_data holds academic records, uploaded-document references, and computed readiness scores; company_data holds recruiter and organisation profiles; job_data holds posting details and eligibility criteria; placement_data holds recorded drive outcomes; notification_data holds per-user notification records; and messages holds the message threads exchanged between roles. Collection names may be adapted during deployment without altering the application logic, since access is mediated entirely through the driver calls described in Section V-E.

D. Hardware and Software Requirements

The application runs comfortably on consumer-grade hardware with at least 4 GB of RAM for local development, a Node.js runtime compatible with the Express.js version in use, and a running MongoDB instance (local or Compass-managed). A stable internet connection is required only for outbound Nodemailer email delivery during OTP verification and notifications; the remainder of the application functions over a local or intranet connection, which is a practical consideration for institutions with constrained external connectivity.

VII. RESULTS AND DISCUSSION

The platform was implemented and functionally exercised across all four roles during development, with particular attention to session integrity across concurrent role logins, correctness of file-upload handling, and the arithmetic of the readiness-score computation against test student records.

• Role Separation: Session-aware middleware correctly restricted each route group to its intended role during testing, and no cross-role route access was observed across the tested scenarios.

• OTP and Session Handling: Email-delivered OTP verification completed reliably within the test environment, and session cookies persisted authentication state correctly across page navigations without requiring re-authentication.

• Upload Handling: Résumé and certificate uploads via express-fileupload were correctly stored and retrievable through the student and recruiter dashboards during testing, including for files at the upper end of the accepted size range.

• Prediction Engine Behaviour: The rule-based readiness score responded predictably to changes in its inputs — reducing correctly as backlog count rose or certification density fell — which is the behaviour expected of a transparent, weighted formula, though this predictability comes at the cost of the pattern-discovery capability a learned model could offer given sufficient historical data.

• Notification Delivery: In-application and email notifications for new postings and shortlist outcomes were delivered without perceptible delay under the tested load, though the current implementation has not been evaluated under the concurrent load of a full institutional cohort.

These observations indicate that the platform meets its core functional objectives — consolidated role-based workflows, secure registration, and a transparent readiness score — under development-scale testing. Two limitations are worth noting explicitly. First, Base64 encoding of credentials, while sufficient to obscure passwords from casual inspection during development, is not a substitute for a salted cryptographic hash and should be replaced before any production deployment handling real student data. Second, because the prediction engine's weights were set manually rather than fit against historical placement outcomes, its scores should be read as a structured, transparent heuristic rather than a statistically validated probability of placement.

VIII. CONCLUSION AND FUTURE SCOPE

This paper presented the Placement Prediction and Management System, a role-based web platform built on Node.js, Express.js, EJS, and MongoDB that consolidates student, recruiter, coordinator, and administrator workflows around a transparent, rule-based placement-readiness score. The design prioritised auditable scoring logic and self-contained

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 4, July 2026
Impact Factor: 8.2

authentication over the additional sophistication of a learned predictive model or a third-party identity provider, a trade-off suited to an initial deployment without a pre-existing historical placement dataset.

Planned future work includes replacing Base64 credential encoding with a salted cryptographic hashing scheme such as bcrypt ahead of any production deployment; collecting outcome-labelled placement data over successive cohorts so that the current rule-based score can be supplemented or replaced by a trained predictive model; extending the recruiter module to support multi-round interview scheduling rather than single-stage shortlisting; adding analytics dashboards that track placement trends across academic years; and conducting a structured usability evaluation with actual coordinators and students to validate the workflow assumptions made during development.

REFERENCES

[1] R. Sharma and A. Verma, "Design and development of a web-based college placement management system," Int. J. Comput. Sci. Eng., 2021.

[2] P. Deshmukh, S. Joshi, and N. Kale, "An automated campus recruitment portal using the MERN stack," Int. J. Adv. Res. Comput. Sci., 2022.

[3] K. Menon and R. Pillai, "A survey of e-recruitment systems and résumé-matching techniques," Int. J. Res. Appl. Sci. Eng. Technol., 2023.

[4] S. Bansal, T. Iyer, and M. Nair, "Predicting student placement outcomes using supervised machine learning," Int. J. Eng. Res. Technol., 2022.

[5] A. Fernandes and D. Costa, "A role-based access control model for academic web applications," Int. J. Comput. Appl., 2021.

[6] V. Rao and H. Shetty, "Session management and authentication strategies in Node.js applications: A comparative study," Int. J. Sci. Res. Sci. Eng. Technol., 2023.

[7] N. Bhatt, "Design considerations for MongoDB schema modelling in role-based web platforms," ResearchGate preprint, 2022.

[8] L. Fernandez et al., "OTP-based two-factor verification for lightweight web authentication," Int. J. Emerg. Technol. Innov. Res., 2021.

[9] M. Prabhu and S. Ghosh, "Automated notification and messaging subsystems in enterprise web applications," Int. J. Comput. Appl., 2022.

[10] J. Correia and P. Almeida, "Rule-based versus learned scoring systems for candidate evaluation: A comparative discussion," Applied Sciences (MDPI), 2023