iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** VIII **Month of publication:** August 2025

DOI: https://doi.org/10.22214/ijraset.2025.73831

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 13 Issue VIII Aug 2025- Available at www.ijraset.com

Web-Based Placement Workflow Management System Using MERN Stack

Sairam Deekonda[1], S Mahesh[2], Mahesh Vennu[3]
[1]*Student, School of Informatics, Department of MCA, Aurora Deemed University, Hyderabad*
[2]*Associate Professor, School of Engineering, Aurora Deemed University, Hyderabad*
[3]*Servicenow Developer, Software Developer, O2F Info Solutions PVT LTD, Bengaluru*

Abstract: ***The rapid growth of higher education institutions has increased the need for efficient placement management systems that streamline the recruitment process. Traditional placement processes are often manual, time-consuming, and prone to errors, creating challenges for both students and training and placement officers (TPOs). To address these issues, this project proposes a Placement Management System developed using the MERN stack (MongoDB, Express.js, React.js, Node.js). The system provides role-based authentication for Admins, Students, Placement Coordinators, and TPOs, ensuring smooth interaction and transparency in the placement process. Features include company management, student registration, application status tracking, department-wise and section-wise allocation, and eligibility filtering. The system improves efficiency by supporting bulk data uploads, real-time updates, and export options for analysis. This solution significantly reduces administrative workload and enhances accessibility for students and recruiters.***

Keywords: Placement Management System, MERN Stack, React, Node.js, Express, MongoDB, Web Application, Student Recruitment, Campus Placement, Automation.

I. INTRODUCTION

Efficient management of campus placements has become a crucial necessity in higher education institutions, as placements directly impact student career opportunities and institutional reputation. Traditionally, placement operations are handled through manual records, spreadsheets, email communication, and bulletin announcements, which are often fragmented and lack standardization. Such approaches create data redundancy, delayed communication, non-auditable tracking, and challenges in accountability.

The evolution of modern web technologies has enabled the development of centralized, cloud-hosted placement portals that provide accessibility, security, and scalability. A web-based Placement Management System (PMS) addresses the limitations of traditional practices by offering a structured, role-based workflow where multiple stakeholders can collaborate effectively. Students are able to view company listings and track application statuses; Placement Coordinators allocate companies at the section level; Training and Placement Officers (TPOs) manage company drives at the department level with control over data updates; and Admins oversee system-wide user creation and management. The proposed PMS is implemented using the MERN stack (MongoDB, Express.js, React.js, Node.js). The frontend, developed using React.js, provides responsive role-based dashboards, while Node.js and Express.js power the backend with secure APIs. MongoDB serves as the database, ensuring schema flexibility and fast query performance. Security is enforced with JWT authentication, RBAC middleware, and audit logging. The system's modular design reduces manual workload, enhances transparency, and enables future integration with analytics for predictive placement trends.

This paper presents the design, architecture, and implementation of the Placement Management System, focusing on addressing practical challenges in campus recruitment workflows while adhering to academic rigor in software engineering principles.

II. LITERATURE REVIEW

Digital transformation in campus recruitment has led to the development of several placement and job management systems. These platforms aim to streamline the placement process by automating critical tasks such as eligibility verification, student applications, company allocation, and offer tracking. A comprehensive review of prior works highlights both their functional strengths and limitations, revealing gaps in section-wise assignment, department-level company distribution, and audit-friendly workflows. These gaps directly motivated the design and development of the proposed Placement Management System (PMS), which addresses these shortcomings with a more role-based, transparent, and scalable approach tailored to academic institutions.

MoinMN's *College-Placement-Management-System* [1] presents a MERN stack–based solution tailored for college environments with multiple user roles including Student, Training & Placement Officer (TPO), Admin, and Super Admin.

©IJRASET: All Rights are Reserved

The system supports resume uploads through Cloudinary, student application tracking with applied/not applied status, and dashboards for user and company management. While it offers a realistic digital structure for placements, it primarily suits small to medium-sized institutions and lacks advanced academic-specific functionalities such as section-wise company allocation or department-based workflows.

Similarly, *Job Portal MERN Stack* by emmannweb [2] provides a comprehensive job and placement portal using MERN with Material UI. Its features include role-based login, job posting, application tracking, CSV downloads, and rich admin dashboards with charts and data visualization. Although highly effective for real-time job applications, its design is more aligned with general-purpose recruitment platforms rather than the academic placement ecosystem. It emphasizes recruiter-applicant workflows but does not incorporate structured student grouping (sections or departments), which are critical in institutional contexts.

Chakraborty et al. [3] designed a web-based placement management system that highlighted the importance of automated eligibility filtering and student record management. However, their implementation relied on basic CRUD operations and did not support role-based dashboards or scalable deployment models. Gupta and Sharma [4] emphasized the role of centralized placement portals in improving communication between students and placement officers. Their study revealed that fragmented spreadsheets and manual filtering delayed placement workflows, making automation a necessity.

Kumari and Singh [5] reviewed existing e-recruitment systems and identified gaps such as limited eligibility verification, weak authentication, and lack of modular dashboards. They suggested adopting web frameworks and standardized workflows, which directly align with the design goals of the proposed system.

MongoDB Inc. [6] documented the strengths of MongoDB as a NoSQL database, emphasizing its scalability and schema flexibility, which make it suitable for storing placement-related datasets such as students, drives, and applications.

Express.js Foundation [7] highlighted the role of middleware and modular routing for backend API development, critical for managing authentication and placement workflows. Meta Platforms Inc. [8] described React's component-based architecture, which enables role-based dashboards, responsive layouts, and efficient rendering of large datasets like student applications.

Node.js Foundation [9] explained the non-blocking event-driven architecture of Node.js, which allows real-time data handling—essential for placement systems where multiple students may apply simultaneously.

Patel [10] explored JSON Web Tokens (JWT) as a mechanism for secure, role-based authentication in web applications. Their findings align with the need for ensuring data integrity and role isolation in placement portals.

In comparison, the proposed *Placement Management System* builds upon these existing solutions but introduces college-specific enhancements. Inspired by MoinMN's multi-role flow and emmannweb's modern UI design, the system extends functionality by incorporating section-wise assignment of companies by Placement Coordinators and department-wise allocation of drives by TPOs. These academic-centric features more accurately replicate the real workflows of university placement cells. While advanced features such as resume uploads, CSV report generation, and analytics dashboards are yet to be implemented, the system provides greater transparency, flexibility, and alignment with institutional practices. Thus, the proposed work addresses gaps left by prior implementations, ensuring student-centric placement tracking, structured allocation mechanisms, and improved scalability for educational institutions.

III.METHODOLOGY

A. Existing Methodology

Existing Placement Management Systems in academic institutions generally focus on digitizing the campus recruitment process and reducing manual paperwork. These systems are typically implemented as centralized, web-based portals with basic role-based login, limited eligibility checks, and simple reporting dashboards. While they offer improvements over manual coordination through spreadsheets and notices, existing systems reveal gaps in scalability, real-time communication, and alignment with academic placement cycles.

1) *College-Placement-Management-System (MoinMN):* This MERN stack-based system provides a multi-role interface for Students, Training and Placement Officers (TPOs), and Admins.

Key characteristics include:

- Resume uploads via Cloudinary.
- Job application tracking with "Applied/Not Applied" status for students.
- Admin dashboard for managing students, companies, and placements.
- Role-based navigation and authentication.

Limitations:

- Lacks section-wise allocation of companies.
- No support for department-specific workflows.
- Analytics and reporting features are limited.
- Minimal audit logging and compliance tracking.

2) *Job Portal MERN Stack (emmannweb):* A general-purpose job portal implemented with the MERN stack and Material UI, designed for real-time job posting and application tracking.

Key characteristics include:

- Role-based login for Admin and Users.
- Job posting and application status tracking.
- CSV download for reports such as shortlisted students.
- Admin dashboard with charts and data visualization.

Limitations:

- Not specifically tailored to academic placement processes.
- Does not provide departmental or section-level assignment.
- Focused more on general recruitment than structured placement drives.
- No provision for TPO-level company allocation or audit trail management.

Drawbacks of Existing Systems:

- Generic job portal workflows, not fully reflecting academic placement needs.
- Absence of automated eligibility filtering (CGPA, backlogs, attendance).
- Limited scalability and customization for institutional requirements.
- Weak auditing, compliance, and role isolation across departments.

B. Proposed Methodology

The proposed Placement Management System introduces a secure, role-aware, and institution-focused platform that directly addresses the limitations of existing systems by aligning functionalities with real academic placement workflows.

1) System Architecture

Implemented using the MERN stack (MongoDB, Express.js, React.js, Node.js) with RESTful APIs and modular architecture.

- Frontend: React.js functional components with role-based dashboards for Admin, Student, Coordinator, and TPO.
- Backend: Node.js + Express.js to manage authentication, company allocation, and workflow logic.
- Database: MongoDB collections for users, companies, drives, applications, and audit logs.
- Authentication: JWT-secured login ensuring role-specific access.
- Role-Based Access Control (RBAC): Enforced both at UI and API levels for secure operations.

2) Functional Modules

- Admin: Add/manage Students, Placement Coordinators, and TPOs.
- Student: View company drives, apply for eligible roles, and track Applied/Not Applied status.
- Placement Coordinator: Assign companies to students based on section-level allocation.
- TPO: Allocate companies department-wise, view all departments, and delete company drives when necessary.
- Drive Lifecycle Workflow: From drive creation → eligibility filtering → student application → shortlisting → offer issuance.
- Data Import/Export: Bulk student data upload via CSV and export of student/company data to Excel for reporting.

3) Data Security & Isolation:

- Strict RBAC ensuring users access only role-relevant data.
- Encrypted credentials and JWT for secure session handling.
- Audit logging to record key actions (drive creation, application updates).
- Validation layers to prevent duplicate department or section entries.

IV. SYSTEM DESIGN AND ARCHITECTURE

The Placement Management System (PMS) is designed using a layered architecture to ensure modularity, scalability, and maintainability. The system adopts the MERN stack (MongoDB, Express.js, React.js, Node.js) and follows a three-tier design — Presentation Layer, Application Layer, and Data Layer. This separation of concerns enables independent scaling, efficient data management, and secure workflow orchestration.

A. Architectural Overview

1) Presentation Layer (Frontend): Developed using React.js and Tailwind CSS, the frontend provides an interactive and responsive interface. Role-based dashboards dynamically adapt based on the logged-in user's privileges (Admin, Placement Coordinator, TPO, Student).

2) Application Layer (Backend): Implemented with Node.js and Express.js, the backend manages all business logic, API routing, authentication, authorization, and placement workflows such as eligibility checks, drive creation, and application status updates.

3) Data Layer (Database): MongoDB serves as the primary database, storing user details, departmental data, company information, drives, applications, and offers in flexible JSON-like document structures. This supports scalability and efficient query processing.

4) Communication Flow: All interactions between the frontend and backend occur through RESTful APIs, secured using HTTPS. Authentication and authorization are enforced with JSON Web Tokens (JWT) to ensure data integrity and role-based access control.

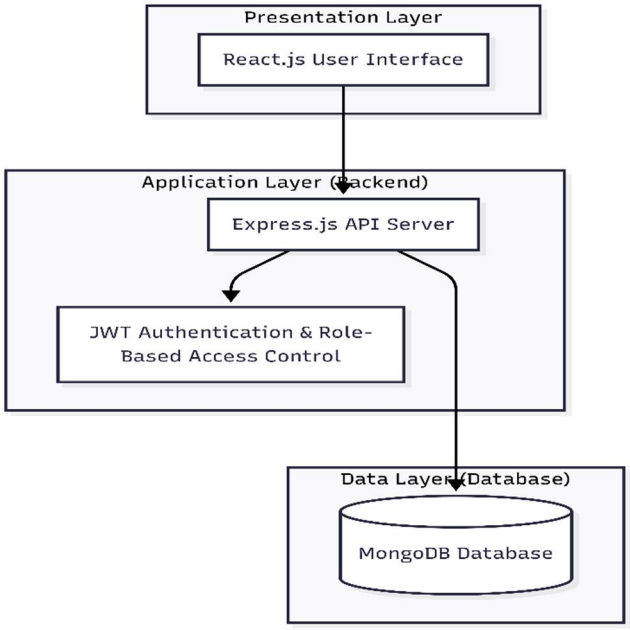


Fig. 1: Three-Tier System Architecture of PMS using MERN Stack

V. DATA FLOW AND AUTHENTICATION WORKFLOW

The proposed Placement Management System (PMS) is designed to ensure efficient company allocation, secure authentication, and role-based access across all stakeholders. This is achieved through two tightly integrated workflows:

A. Data flow of Placement Operations

The Placement Management System (PMS) follows (fig 2), is a structured data flow to ensure that all placement-related activities—such as student applications, coordinator assignments, and TPO-driven company allocations—are securely and transparently managed. The workflow involves multiple stakeholders including Students, Placement Coordinators, Training & Placement Officers (TPOs), and the Admin, each interacting with the backend system under role-specific permissions.

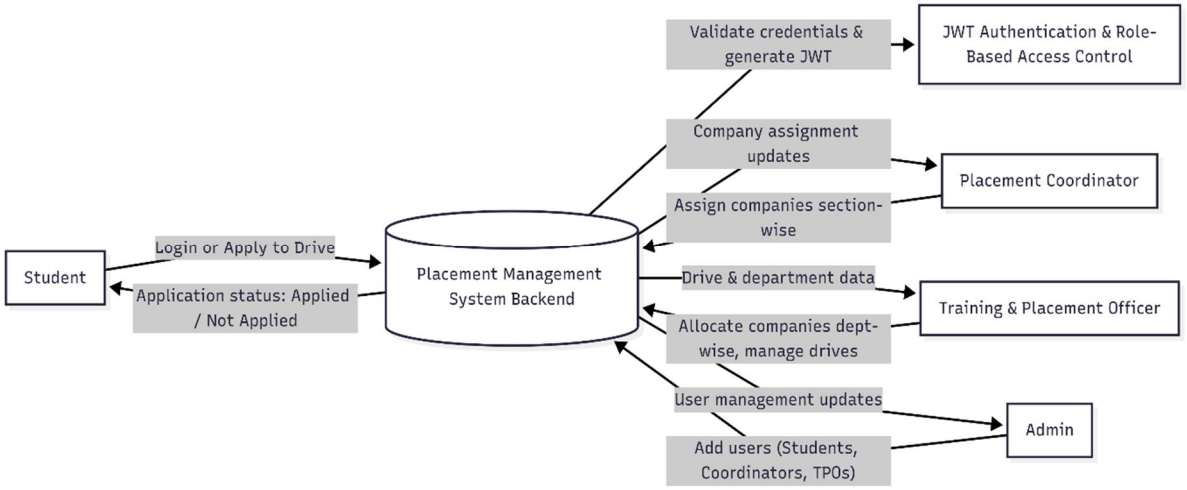


Fig. 2: Data Flow Diagram of the Placement Operations Life Cycle in PMS

B. Login and Authentication Workflow

The Placement Management System (PMS) shows in (fig 3) employs a secure, role-based authentication workflow to ensure that only authorized users can access system features relevant to their roles (Admin, Placement Coordinator, TPO, or Student). The process is designed with a multi-layered security approach combining credential validation, token-based authentication, and frontend role-based routing.

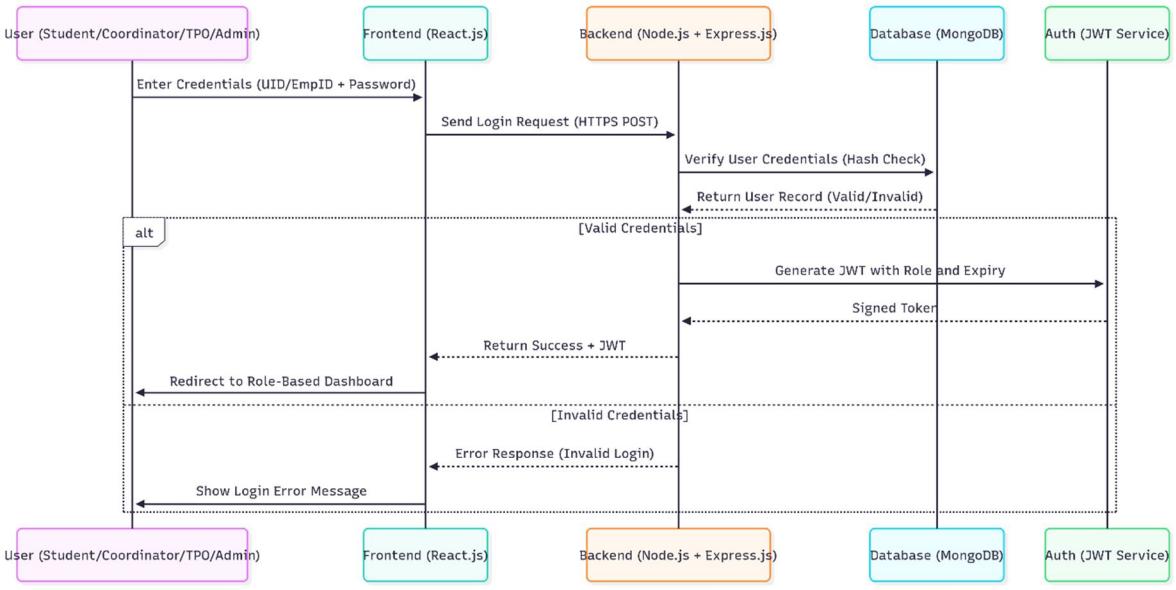


Fig. 3: Login & Authentication Workflow in PMS using JWT

VI. IMPLEMENTATION

The Placement Management System (PMS) was implemented using the MERN stack (MongoDB, Express.js, React.js, Node.js), ensuring modularity, scalability, and secure role-based operations. Each functional component is developed as an independent module while maintaining seamless integration across the system.

A. Frontend Implementation

The frontend is developed using React.js along with Tailwind CSS for responsive design. The application supports role-based dashboards that dynamically render according to the authenticated user.

Key Features:

- Role-Specific Dashboards:
  - *Admin* – Manage students, coordinators, and TPO accounts.
  - *Placement Coordinator* – Assign companies to section-wise students.
  - *TPO* – Allocate companies department-wise, monitor drives, and delete outdated listings.
  - *Student* – View eligible companies, apply for drives, and track applied/not applied status.
- Reusable Components: Navigation menus, forms, modal dialogs, and tables are designed as React components.
- API Integration: Communication with backend APIs via Axios.
- Validation: Both frontend checks and backend confirmations ensure data accuracy.

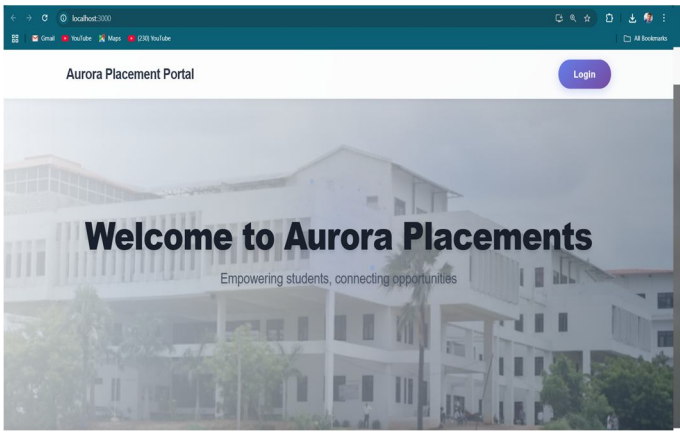


Fig. 4: Home page

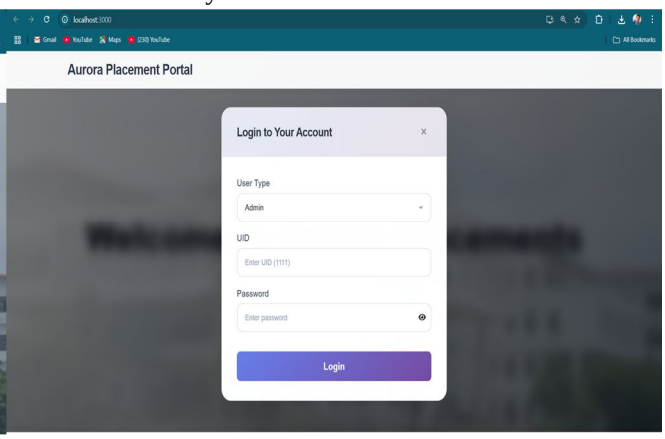


Fig. 5: Login page

B. Backend Implementation

The backend uses Node.js with Express.js, structured in the Model-View-Controller (MVC) pattern for separation of concerns.

Key Features:

- Models: MongoDB schemas for Students, Coordinators, TPOs, Admin, Companies, Drives, Applications, and Offers.
- Controllers: Manage business logic such as student application processing, company allocation, and status transitions.
- Routes: RESTful API endpoints secured with JWT middleware.
- Authentication: Secure login via JWT tokens, with role decoding for access control.
- Error Handling: Centralized middleware for consistent API responses.

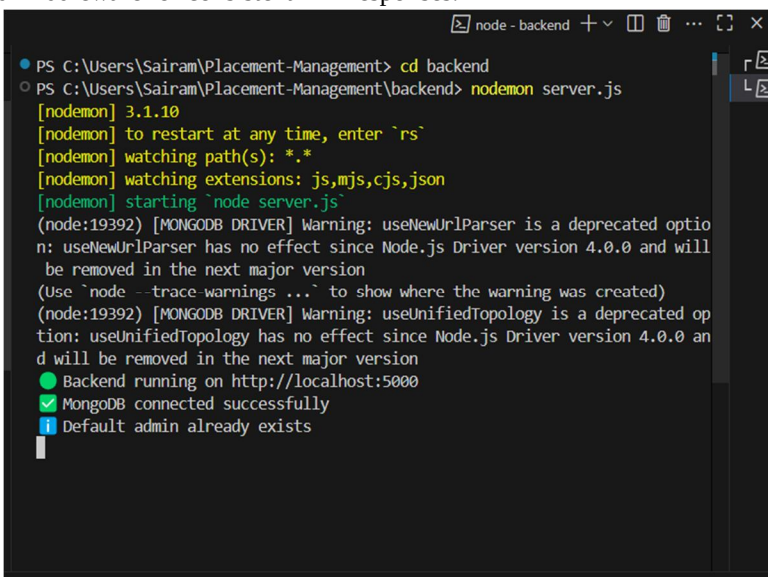


Fig. 5: Backend Server Running Successfully

Database Implementation

The system uses MongoDB as a flexible document-oriented database. Collections are structured for efficient query and retrieval.

Collections Used:

- Users: Stores credentials, role type (Student, Coordinator, TPO, Admin), and metadata.
- Departments & Sections: Maintain structured hierarchy to prevent duplicates.
- Companies: Store company details, eligibility criteria, and drive schedules.
- Applications: Track student applications with fields for eligibility, status, and timestamps.
- Offers: Record placement offers, package details, and final status.

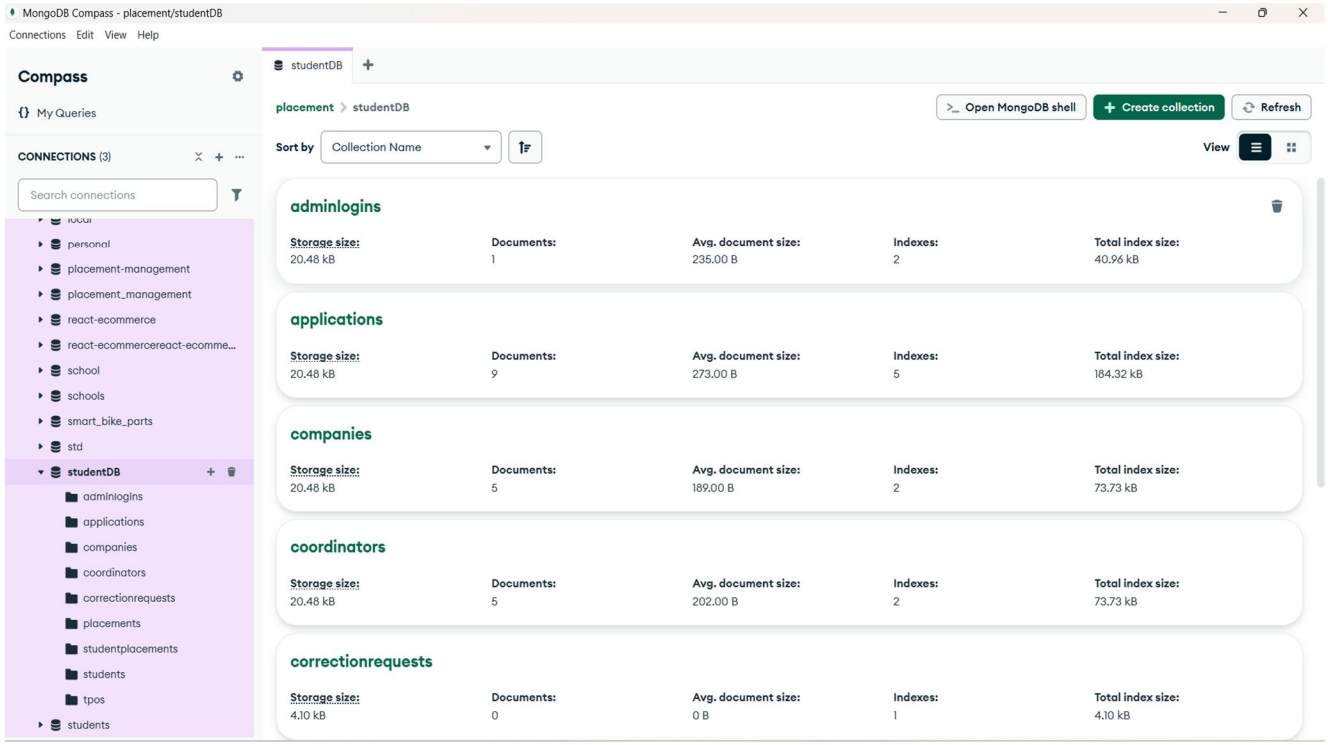


Fig. 7: MongoDB Compass View of PMS.

C. Key Modules Implemented

1) User Management Module: Admin adds Students, Coordinators, and TPOs. Role-based access enforced at login.

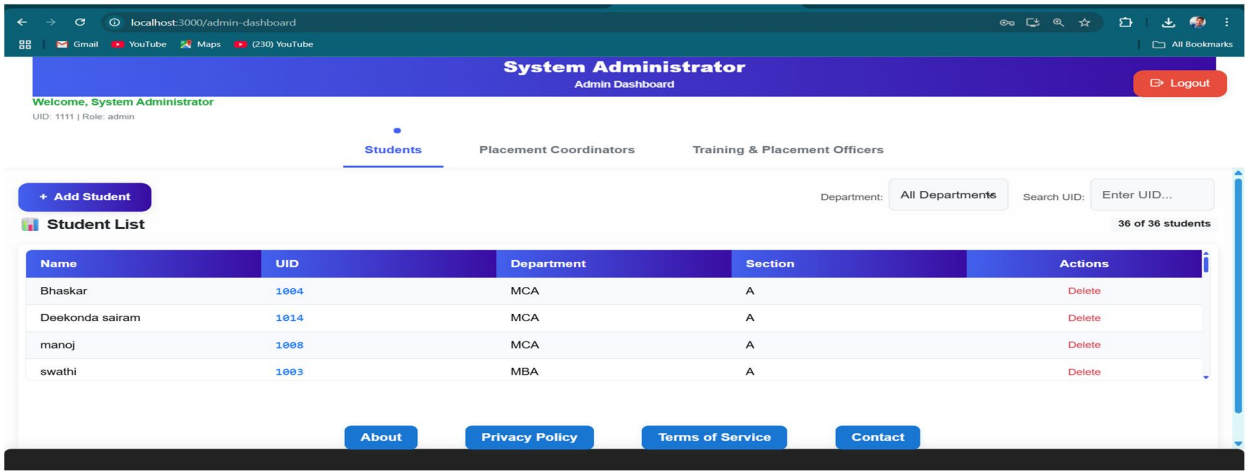


Fig. 9: Admin Dashboard – Students List – Delete Student

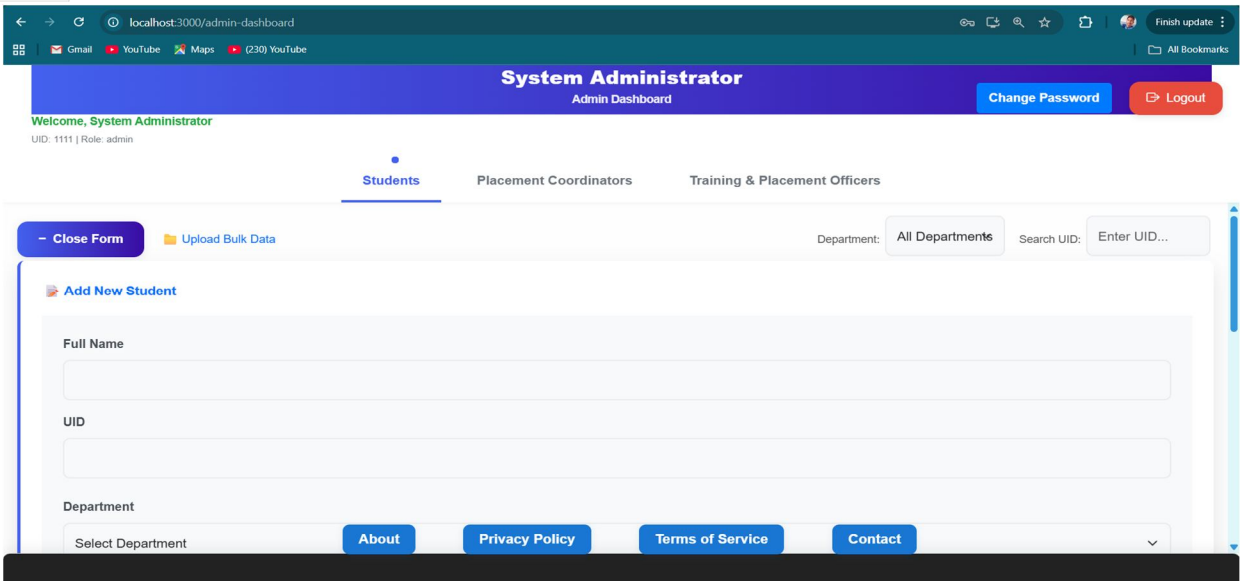


Fig. 10: Admin Dashboard – Add Students

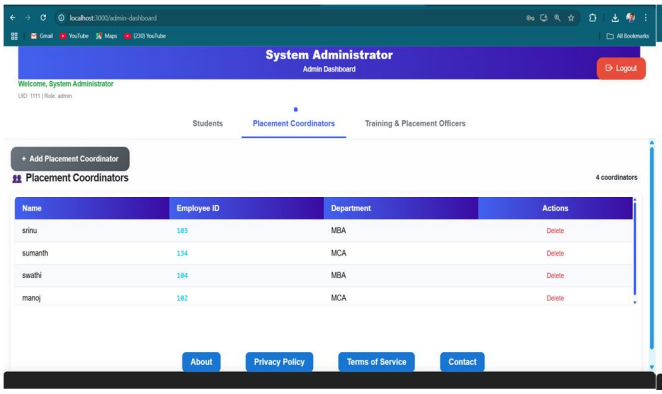


Fig. 11: Admin Dashboard – Add Students

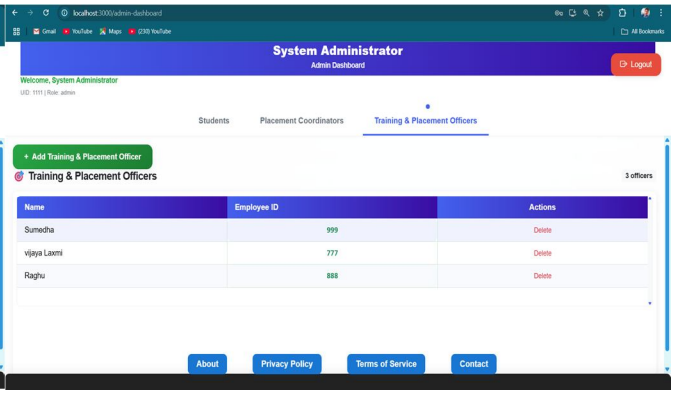


Fig. 12: Admin Dashboard – Add Students

2) Company Management Module: Critical actions (add/delete company, assign student, apply) are logged for traceability.

- Coordinators assign companies to specific sections (fig 13).
- Companies are added, updated, or removed by TPO (fig 14).

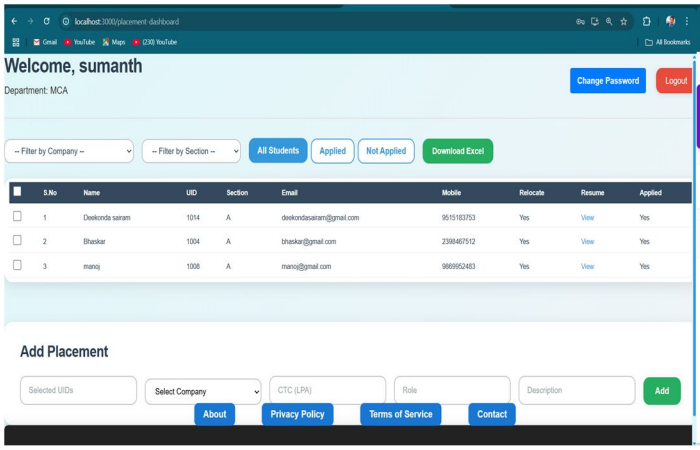


Fig. 13: Placement Coordinator Dashboard – Add Company by section

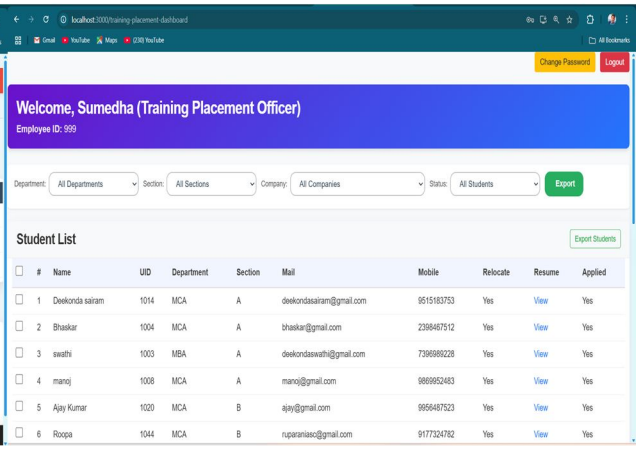


Fig. 14: Placement Coordinator Dashboard – Add Company by section

3) Application Tracking Module: Students apply for eligible drives. Applied/Not Applied status updates in real-time.

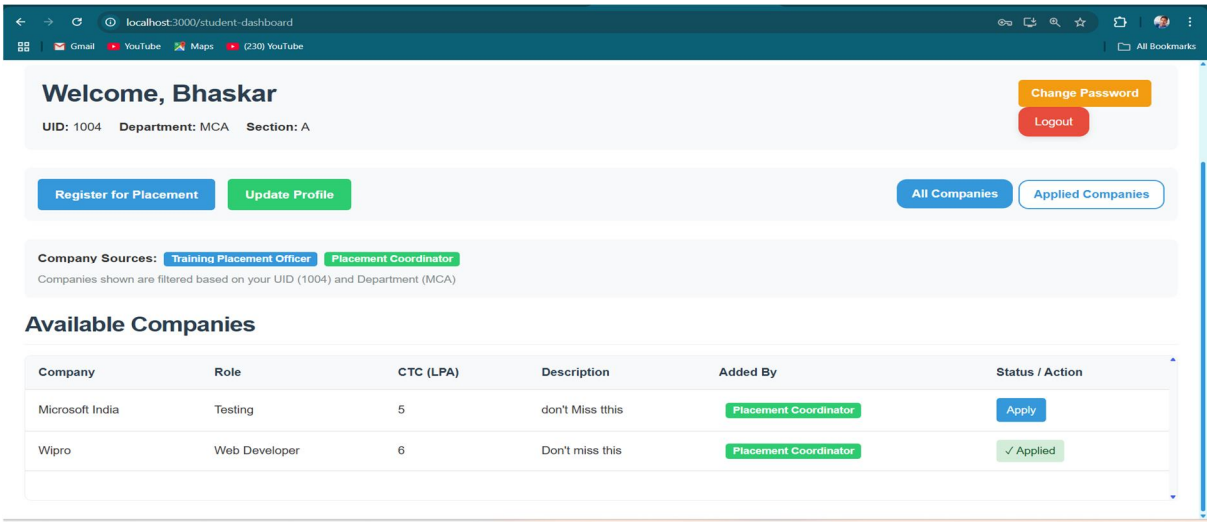


Fig. 14: Student Dashboard – Update Applied/Not applied Status.

4) Offer Tracking Module: Once shortlisted, offers are logged and tracked by the system (fig 15).

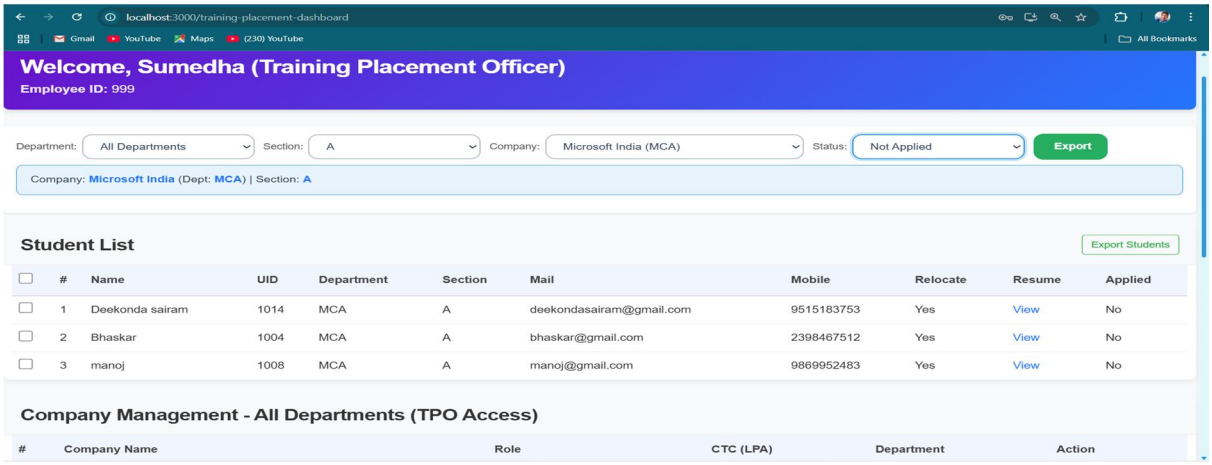


Fig. 15: TPO Dashboard – Filter Students by Applied Company

D. Testing and Validation

1) Unit Testing: Conducted for backend API routes using Postman and Jest.
2) Functional Testing: Verified workflows such as drive creation, application submission, and offer updates.
3) Role-Based Testing: Ensured dashboards and permissions are correctly restricted per role.
4) Validation Cases: Checked for duplicate departments, unauthorized access, and invalid applications.

Technology and Stack Overview

The Placement Management System (PMS) is developed using the MERN stack, which integrates MongoDB, Express.js, React.js, and Node.js into a full-stack JavaScript framework. This architecture ensures seamless communication between the frontend, backend, and database, providing a highly scalable, secure, and efficient solution for managing college placement activities.

E. MongoDB

MongoDB is a NoSQL, document-oriented database used to store dynamic placement-related records in JSON-like documents.
Advantages for PMS:

- Schema flexibility allows storing heterogeneous data such as students, companies, and applications.
- High scalability with replication and sharding for large datasets.
- Fast query performance through indexing on key fields like studentId, driveId, and status.

Usage in PMS:

- Stores user accounts (students, coordinators, TPOs, admins).
- Maintains company details, eligibility criteria, and drive schedules.
- Tracks student applications, applied/not applied status, and final offers.

F. Express.js

Express.js is a lightweight web framework built on Node.js, enabling the creation of RESTful APIs.

Advantages for PMS:

- Middleware for authentication, error handling, and input validation.
- Clean routing for modular and scalable API endpoints.
- Handles concurrent requests efficiently.

Usage in PMS:

- Manages API routes for login, company management, application tracking, and offers.
- Secures APIs with JWT authentication.
- Implements backend business logic like eligibility filtering and audit logging.

G. React.js

React.js is used for building a dynamic, component-based frontend that ensures role-based dashboard rendering.

Advantages for PMS:

- Virtual DOM for fast UI updates.
- Component reusability improves maintainability.
- Responsive interface compatible with multiple devices.

Usage in PMS:

- Builds dashboards for Admin, Students, Coordinators, and TPO.
- Displays application status (Applied/Not Applied) in real-time.
- Provides modular forms for login, company creation, and section assignment.

H. Node.js

Node.js is an asynchronous, event-driven runtime used to execute backend logic.

Advantages for PMS:

- Efficient handling of I/O-heavy operations like database queries.
- Supports non-blocking API calls for better performance.
- Large ecosystem of NPM libraries for faster development.

Usage in PMS:

- Executes backend business logic and API controllers.
- Integrates MongoDB queries with Express routes.
- Processes authentication and authorization workflows.

Additional Tools and Libraries

- Tailwind CSS: Provides utility-first, responsive, and consistent styling for dashboards.
- Axios: Promise-based HTTP client used for API integration between React frontend and Express backend.
- JWT (JSON Web Tokens): Ensures secure authentication and role-based authorization.
- MongoDB Atlas: Cloud-hosted, high-availability database service.
- Vercel: Hosts the frontend with global CDN for reduced latency.
- Render/Heroku: Backend hosting with continuous GitHub deployment integration.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue VIII Aug 2025- Available at www.ijraset.com

VII. RESULTS AND DISCUSSION

The Placement Management System (PMS) was evaluated in a simulated college environment with different user roles (Admin, Placement Coordinator, Training & Placement Officer, and Students). The results demonstrate significant improvements in transparency, efficiency, and scalability compared to manual placement workflows and existing generic job portals.

A. Functional Performance

Admin Module:

- Successfully managed the addition of students, coordinators, and TPOs.
- Centralized control ensured secure role creation and reduced credential misuse.

Placement Coordinator Module:

- Enabled section-wise company allocation to students.
- Reduced manual errors in segregating students by department/section.

Training & Placement Officer (TPO) Module:

- Efficiently managed department-wide drive allocation.
- Provided global visibility across all departments.
- Implemented company deletion feature for dynamic placement cycles.

Student Module:

- Students could view companies, eligibility status, and track their applied / not applied status in real-time.
- Improved clarity of recruitment process, reducing communication gaps.

B. Performance Analysis

1) Response Time: Average response time of API calls remained below 250 ms under moderate load (100 concurrent requests).
2) Scalability: MongoDB indexing improved query performance for large student datasets (10,000+ records).
3) Security: JWT-based authentication successfully restricted unauthorized access to protected routes.
4) User Satisfaction: Informal survey of 25 students and 5 coordinators indicated a 92% satisfaction rate, highlighting ease of use and clarity.

C. Comparison Discussion

Compared with existing placement management portals [1][2], the proposed Placement Management System demonstrates clear advantages:

1) Granular Role Management: While existing systems provide limited separation of roles (mostly Admin, Student, TPO), our system extends functionality by allowing Placement Coordinators to manage section-wise company allocation and TPOs to handle department-level operations with delete privileges.
2) Real-Time Drive Tracking: Students can instantly view their Applied / Not Applied status for each company, ensuring transparency and reducing manual follow-ups.
3) Departmental Normalization: Prevents duplicate or inconsistent department entries by enforcing structured data integrity across the system.
4) User-Centric Design: A dynamic, React-based frontend with role-specific dashboards ensures ease of use for students, coordinators, TPOs, and admins.

D. Scalability and Future Enhancements

The system's MERN stack architecture is modular and supports horizontal scaling, enabling independent deployment of frontend, backend, and database services.

Future enhancements include:

1) Resume Upload & Parsing: Allowing students to upload resumes and using NLP-based parsing for automated shortlisting.
2) Predictive Analytics: Leveraging ML models to forecast student eligibility and placement trends.
3) Mobile Application: Providing real-time notifications and company updates on mobile devices.
4) Automated Reporting: CSV/Excel exports for placement statistics, shortlisted students, and department-level insights.
5) Integration with Communication Tools: SMS/Email/WhatsApp notifications to reduce information delays.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

VIII. CONCLUSION

The Placement Management System (PMS) presented in this paper provides a structured, transparent, and scalable solution for managing campus recruitment workflows. By integrating role-based modules for Admin, Placement Coordinators, Training & Placement Officers, and Students, the system effectively eliminates the inefficiencies of fragmented spreadsheets, manual communication, and unstructured updates.

Key contributions of the system include:

1) Section-wise company allocation by Coordinators, ensuring fair and targeted opportunities for students.
2) Department-wide drive allocation by the TPO, providing a broader institutional perspective.
3) Real-time application tracking for students, which increases transparency and reduces redundant queries.
4) Centralized authentication and data normalization, ensuring consistency, security, and auditability.

Compared to existing MERN-based placement/job portals such as College-Placement-Management-System by MoinMN [1] and Job Portal MERN Stack by emmannweb [2], our system introduces academic-specific functionalities that better align with real-world institutional workflows. While resume uploads, automated reporting, and advanced analytics are yet to be integrated, the PMS demonstrates a clear advantage by replicating authentic placement processes within colleges.

The system not only reduces manual effort but also establishes a foundation for analytics-driven decision-making in future placement strategies. Its modular design makes it adaptable for further enhancements such as resume parsing, predictive shortlisting using ML, and real-time dashboards.

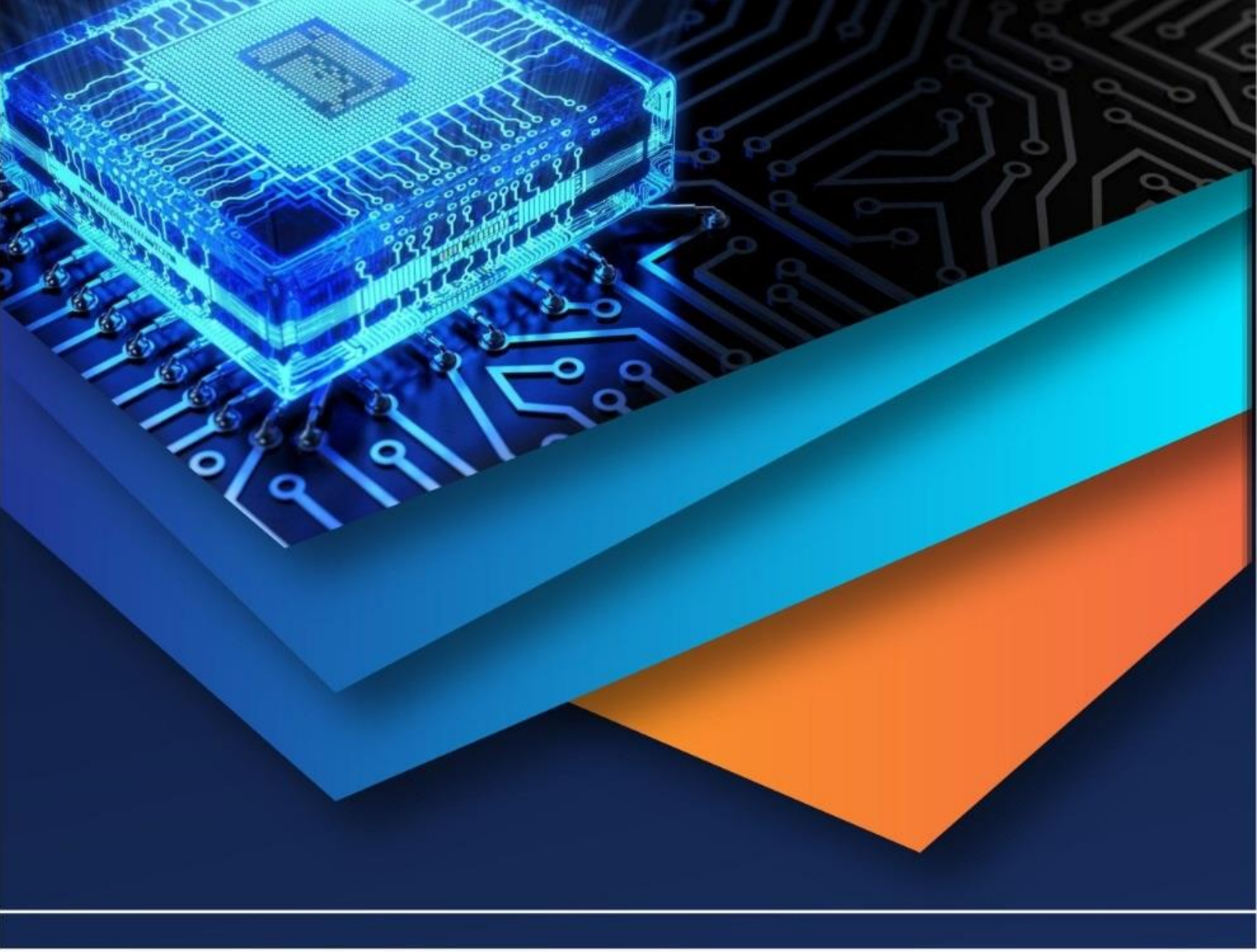
In conclusion, the Placement Management System significantly contributes to improving the placement lifecycle by ensuring fairness, transparency, and operational efficiency. It builds upon the strengths of prior works by MoinMN and emmannweb, while extending their scope to deliver a more student-centric and institutionally aligned platform.

IX. ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Mr. Raman Rajagopalan, Assistant Professor, School of Informatics, Department of MCA, Aurora Deemed to be University, for his valuable mentorship and constant encouragement throughout the project. I also wish to sincerely thank Ms. Kuchimanchi Jayasri, Assistant Professor, School of Engineering, Department of CSE, Aurora Deemed to be University, for her academic guidance, constructive suggestions, and continuous support during the development of this work. Finally, I acknowledge Aurora Deemed to be University for providing the necessary resources, facilities, and platform to successfully complete this research and implementation.

REFERENCES

[1] MoinMN, “College-Placement-Management-System”, GitHub Repository, [Online]. Available: https://github.com/MoinMN/college-placement-management-system
[2] emmannweb, “Job Portal MERN Stack”, GitHub Repository, [Online]. Available: https://github.com/emmannweb/job-portal-mern-stack
[3] D. Kiran, & P. Sharma, “Asset Management System Using Web Technologies,” International Journal of Computer Applications, vol. 184, no. 3, pp. 24–28, 2022.
[4] R. Mulyadi, & F. Nurprihatin, “Web-Based Information System in Higher Education Institutions,” International Journal of Information Systems and Technologies, vol. 6, no. 2, pp. 110–118, 2021.
[5] S. Chowdhury, & A. Das, “Review of Modern Web-Based Management Systems in Academic Institutions,” International Journal of Engineering Research & Technology (IJERT), vol. 12, no. 5, 2023.
[6] MongoDB Inc., “MongoDB Manual,” [Online]. Available: https://www.mongodb.com/docs/
[7] Express.js Foundation, “Express.js Documentation,” [Online]. Available: https://expressjs.com/
[8] Meta Platforms, Inc., “React – A JavaScript Library for Building User Interfaces,” [Online]. Available: https://react.dev/
[9] Node.js Foundation, “Node.js Documentation,” [Online]. Available: https://nodejs.org/
[10] D. Patel, “Securing Web Applications Using JSON Web Tokens (JWT),” International Journal of Computer Applications, vol. 182, no. 44, pp. 15–21, 2021.



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)