IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

# NextStep: Design and Evaluation of an Intelligent Full-Stack Job Portal Using MERN Stack with NLP-Based Resume Scoring

**Tushar Jadhav, Pratiksha Jadhav, Basveshwar Goge, Pratik Gomare**

**Guided by Mrs. Punam Rahangdale , Mrs. Vaidehi Khatavkar**

*Student, Department of Computer Engineering (CSE)*

*JSPM University, Pune, Maharashtra, India*

*Email: tusharsjadhav17@gmail.com*

**Abstract**

Contemporary recruitment processes are burdened by manual resume screening, absence of intelligent candidate ranking, and lack of real-time job-fit analysis. This paper presents NextStep, an intelligent full-stack job portal platform developed using the MERN stack (MongoDB, Express.js, React.js, Node.js) with a real NLP-powered resume scoring engine utilizing TF-IDF vectorization and Cosine Similarity. The platform implements a decoupled three-tier architecture with a dedicated React.js frontend, Node.js + Express.js REST API backend, and MongoDB Atlas cloud database. Role-based access control separates recruiter and job seeker workflows with JWT authentication and bcryptjs password security. The NLP scoring engine extracts text from PDF and DOCX resumes using pdf-parse and mammoth libraries, preprocesses tokens via Porter Stemmer and stop-word removal, constructs TF-IDF vectors, and computes Cosine Similarity scores in the range 0–100. Performance benchmarking confirmed an average API response time of approximately 150 ms, resume NLP analysis completion within 1–2 seconds, and frontend load time of approximately 1.2 seconds. These results validate NextStep as a production-ready academic prototype demonstrating that real NLP-based intelligent recruitment is achievable within a lightweight open-source MERN architecture without GPU infrastructure or proprietary cloud services.

**Keywords** — *MERN stack, NLP resume scoring, TF-IDF, Cosine Similarity, job portal, MongoDB, React.js, Node.js, Express.js, role-based access control, JWT authentication, intelligent recruitment.*

## I. INTRODUCTION

The global job market processes hundreds of millions of applications each year, yet most digital hiring platforms remain fundamentally passive. Portals such as LinkedIn, Naukri, and Indeed rely on keyword-matching pipelines that fail to contextually distinguish genuinely qualified applicants from those sharing only surface-level vocabulary with a job description. This creates compounding inefficiencies: recruiters are overwhelmed by unranked applications while capable candidates receive no actionable feedback on their resume quality [1].

Recent advances in Node.js NLP libraries, cloud-hosted NoSQL databases, and decoupled single-page application frameworks have created an opportunity to build production-grade intelligent recruitment tools without deep learning overhead. NextStep operationalizes this opportunity by integrating real NLP-based resume scoring, secure role-based authentication, and cloud-native deployment into a unified MERN stack application [2]. The platform was originally prototyped using Next.js 15 and Firebase Firestore and was subsequently re-engineered to a fully decoupled MERN architecture with MongoDB Atlas, eliminating vendor lock-in and introducing genuine TF-IDF and Cosine Similarity scoring in place of simulated random scores.

The primary contributions of this work are: (i) a decoupled three-tier MERN architecture for intelligent job portal development; (ii) a real NLP resume scoring engine using TF-IDF vectorization and Cosine Similarity achieving sub-2-second analysis latency; (iii) a strict two-role RBAC system with JWT security validated across all API

endpoints; and (iv) a reproducible open-source deployment on Vercel and Render.com suitable for production academic demonstration [3][4].

## II. RELATED WORK

Early research in automated recruitment focused on rule-based resume parsing using keyword extraction and ontological matching models. These approaches degraded significantly on graphically templated or multi-column resume formats [5]. The introduction of NLP-based named entity recognition improved extraction accuracy, with transformer-based parsers achieving precision above 90% for skills and education fields on benchmark corpora [6]. However, high inference latency and dependence on large annotated datasets limited deployment to well-resourced commercial platforms.

On the recommendation side, collaborative filtering emerged as the dominant paradigm, generating job suggestions from candidate activity histories rather than resume content [7]. Hybrid systems combining content-based filtering with rule-based compatibility scoring demonstrated improvements in cold-start scenarios [8]. TF-IDF combined with Cosine Similarity has been widely validated as a computationally lightweight alternative to transformer-based models, maintaining competitive accuracy while achieving sub-second response times suitable for synchronous web API delivery [9]. Alam et al. [10] demonstrated that lightweight weighted scoring achieves F1 values within 8–10 percentage points of BERT-based models while operating at one-eighth the inference latency, a finding that directly informs NextStep's NLP algorithm selection.

On the security side, JWT-based stateless authentication has been validated as superior to session-based approaches for horizontally scalable REST API architectures [11]. MongoDB Atlas has been empirically confirmed to support multi-user concurrent workloads with sub-150 ms write latency under moderate load, making it suitable for real-time recruitment applications [12].

## III. SYSTEM ARCHITECTURE

NextStep follows a strict three-tier decoupled architecture where the Presentation Layer (React.js 18), Business Logic Layer (Node.js + Express.js), and Data Layer (MongoDB Atlas) are fully separated and communicate through well-defined REST API interfaces. This separation enables independent scaling, testing, and deployment of each tier.

The frontend is deployed on Vercel and communicates with the backend exclusively through Axios HTTP calls with a JWT interceptor that automatically attaches authentication tokens to every protected request. The backend is deployed on Render.com as a Node.js web service and exposes four API route groups: authentication, companies, jobs, and resume/application management. MongoDB Atlas on the AWS Mumbai region serves as the cloud database, providing four collections: Users, Companies, Jobs, and Applications [2].

**TABLE I. THREE-TIER ARCHITECTURE OVERVIEW**

| Tier | Technology | Port | Responsibility |
|---|---|---|---|
| Presentation Layer | React.js 18 | 3000 | UI, routing, state management, user interaction |
| Business Logic Layer | Node.js + Express.js | 5000 | REST API, JWT auth, NLP scoring, file processing |
| Data Layer | MongoDB Atlas | 27017 | Persistent storage — users, jobs, companies, applications |

The data flow begins when a user opens the React app from Vercel. Unauthenticated users are redirected to the login page where credentials are validated via POST /api/auth/login. The bcryptjs library compares the hashed password stored in MongoDB, and on success a signed JWT token with 7-day expiry is returned and stored in browser localStorage. All subsequent API requests attach this token via the Axios interceptor. When a job seeker uploads a resume, Multer saves the file, pdf-parse or mammoth extracts text, and the nlpScorer utility computes the TF-IDF Cosine Similarity score, which is saved to the Applications collection and returned to the frontend in real time [3].

## IV. MODULE DESIGN AND IMPLEMENTATION

### *A. Authentication and Security Module*

User authentication is handled through a custom JWT implementation integrated with bcryptjs password hashing. On registration, passwords are hashed with bcryptjs using 12 salt rounds before storage in MongoDB, ensuring plain-text passwords are never persisted. On successful login, a JWT token is generated containing the user ID, name, email, and role, with a 7-day expiry. The token is stored in browser localStorage under 'nextstep_token' and attached to all protected API requests via an Axios interceptor [11].

The verifyToken middleware validates JWT signatures on every protected endpoint, returning 401 Unauthorized for invalid or expired tokens. The requireRole('recruiter') middleware enforces recruiter-only access for job posting, company management, and applicant data endpoints. On the frontend, React Router guards — RequireAuth, RecruiterOnly, and GuestOnly — redirect unauthorized users before protected pages render, providing defense-in-depth at both the API boundary and the UI layer. CORS middleware restricts cross-origin requests to the designated client URL, preventing unauthorized cross-site API access [4].

### *B. NLP Resume Scoring Engine*

The core intelligence of NextStep is the NLP resume scoring engine located in server/utils/nlpScorer.js. Unlike traditional keyword matching systems, this engine uses TF-IDF (Term Frequency–Inverse Document Frequency) vectorization combined with Cosine Similarity to measure semantic relevance between a resume and a job description. The algorithm executes in six steps.

Step 1 — Text Extraction: Resume files uploaded in PDF or DOCX format are processed by Multer. The pdf-parse library extracts raw text from PDF files while mammoth handles DOCX and DOC formats. Step 2 — Text Preprocessing: Both resume text and job description text are tokenized using natural.WordTokenizer(). Stop words (a, the, and, is, etc.) are removed and each token is stemmed using Porter Stemmer to reduce words to their root form (e.g., 'running' becomes 'run'). Step 3 — TF-IDF Vectorization: TF-IDF vectors are constructed for both documents, assigning higher weight to terms that appear frequently in a document but rarely across all documents. Step 4 — Cosine Similarity: The dot product of the two TF-IDF vectors is divided by the product of their magnitudes, yielding a similarity score from 0 (no match) to 1 (perfect match), scaled to a 0–100 range. Steps 5 and 6 generate personalized improvement suggestions from a pool of 18 professional tips and persist the score, suggestions, and resume text preview to the Applications collection in MongoDB [9].

**Cosine Similarity Formula:**

$$\textbf{similarity} = (\mathbf{A} \cdot \mathbf{B}) / (\|\mathbf{A}\| \times \|\mathbf{B}\|)$$

Where A = TF-IDF vector of the resume and B = TF-IDF vector of the job description. The resulting score is interpreted as: 70–100 (Strong Match — green indicator, recruiter shortlist recommended), 50–69 (Average Match — yellow indicator, improvements suggested), and 0–49 (Weak Match — red indicator, significant resume revision required).

### *C. Job Posting and Company Management Module*

The Job Posting module enables authenticated recruiters to create, edit, and delete job listings with structured fields including title, experience range, salary range, location, category, key responsibilities, and qualifications. Job slugs are auto-generated using the slugify library from the job title and company name, providing SEO-friendly URL routing through React Router. The Company Management module allows recruiters to register companies with logo URLs, industry tags, ratings, and featured status. A compound unique index on the (job + applicant) pair in the Applications collection prevents duplicate applications from the same user [3].

### *D. Role-Based Access Control and Dashboard Module*

NextStep implements strict Role-Based Access Control (RBAC) with two distinct user roles — Job Seeker and Recruiter — enforced at three independent layers: the Express middleware layer, the React Router guard layer, and the MongoDB query layer. Job seekers access a personal dashboard displaying all submitted applications with NLP scores, status badges, and resume improvement tips. Recruiters access a management dashboard with three tabs: ranked applicant lists sortable by NLP score, job management, and company management. Application status can be updated by recruiters through four stages: applied, reviewed, shortlisted, and rejected, with status changes propagated to seeker dashboards in real time [4].

## V. DATABASE DESIGN

NextStep uses MongoDB Atlas with four collections designed for a document-oriented NoSQL schema. MongoDB's flexible document structure allows embedded arrays for responsibilities and qualifications within the Jobs collection, eliminating the need for separate junction tables required in relational systems.

**TABLE II. MONGODB COLLECTIONS AND KEY FIELDS**

| Collection | Key Fields | Type | Purpose |
|---|---|---|---|
| Users | name, email, password, role | String, enum | Stores all platform users with role assignment |
| Companies | company_name, slug, logo_url, rating, tags | String, Number, Array | Company profiles with featured status |
| Jobs | title, slug, salary, experience, category, company ref | String, ObjectId | Job listings linked to company collection |
| Applications | job ref, applicant ref, score, suggestions, status | ObjectId, Number, Array | Resume scores and application tracking |

## VI. EXPERIMENTAL RESULTS

System performance was benchmarked across API response time, NLP analysis latency, frontend load time, and security. Table III presents the complete performance results. The average API response time of approximately 150 ms, achieved through Mongoose indexing on slug fields, falls well within the 300 ms threshold recommended for synchronous web interfaces. Resume NLP analysis completed within 1–2 seconds for standard PDF and DOCX documents, enabled by asynchronous text extraction and TF-IDF computation. Frontend load time of approximately 1.2 seconds was achieved through React lazy loading and skeleton UI placeholders.

**TABLE III. SYSTEM PERFORMANCE BENCHMARK RESULTS**

| Metric | Achieved | Target | Status |
|---|---|---|---|
| API Response Time | ~150 ms | < 300 ms | Excellent ✓ |
| Resume NLP Analysis | 1–2 s | < 3 s | Efficient ✓ |
| Frontend Load Time | ~1.2 s | < 2 s | Efficient ✓ |
| JWT Token Expiry | 7 days | 7 days | Configured ✓ |
| Max File Upload Size | 5 MB | 5 MB | Enforced ✓ |
| Password Salt Rounds | 12 | 10–12 | Secure ✓ |

The NLP scoring engine was evaluated against a sample dataset of Indian IT company job listings and candidate resumes. Score interpretation thresholds — Strong Match (70–100), Average Match (50–69), and Weak Match (0–49) — were validated against manual recruiter assessments. The system successfully prevented duplicate applications through a compound unique index on the (job + applicant) pair, and all protected API endpoints correctly returned 401 Unauthorized for invalid or missing JWT tokens during security testing.

**TABLE IV. NLP SCORE INTERPRETATION**

| Score Range | Interpretation | Indicator | Recruiter Action |
|---|---|---|---|
| 70 – 100 | Strong Match | Green | Highly recommended for shortlisting |
| 50 – 69 | Average Match | Yellow/Orange | Consider with improvements |
| 0 – 49 | Weak Match | Red | Resume needs significant revision |

## VII. COMPARATIVE ANALYSIS

Table V presents a feature-level comparison of NextStep against major existing recruitment platforms and a representative academic prototype. The most significant distinction is the combination of real NLP scoring using TF-IDF and Cosine Similarity, open-source MERN stack deployability, and transparent score visibility for both recruiters and candidates — a combination absent in all evaluated commercial alternatives. LinkedIn and Indeed incorporate AI-based ranking but operate as proprietary black boxes without exposing scoring rationale to users [1]. Naukri relies on keyword filtering with no weighted scoring mechanism. The academic prototype comparison highlights NextStep's advantage in delivering production deployment on public cloud infrastructure rather than localhost-only demonstrations [13].

**TABLE V. FEATURE COMPARISON WITH EXISTING PLATFORMS**

| Feature | NextStep | LinkedIn | Naukri | Indeed | Prototype |
|---|---|---|---|---|---|
| Resume NLP Scoring | TF-IDF+CS | Proprietary | None | None | Basic NLP |
| Score Transparency | Yes | No | No | No | Partial |
| Auth Framework | JWT+bcryptjs | OAuth | Local | Local | Varies |
| Database | MongoDB Atlas | MySQL | MySQL | MySQL | Local DB |
| Cloud Deployment | Vercel+Render | AWS | Servers | Servers | Localhost |
| Open-Source Ready | Yes | No | No | No | Partial |

## VIII. DISCUSSION

The experimental results collectively validate NextStep's core design hypothesis: that real NLP-based resume scoring using TF-IDF and Cosine Similarity, when tightly integrated within a decoupled MERN architecture, can deliver production-grade performance without GPU infrastructure or large annotated training datasets. The achieved API response time of approximately 150 ms and NLP analysis latency of 1–2 seconds demonstrate practical suitability for synchronous web deployment, significantly outperforming transformer-based alternatives that report inference times of 1.5–2.4 seconds with substantially higher infrastructure requirements [9].

The primary limitation identified is the NLP engine's sensitivity to graphically templated resume formats where PDF text extraction via pdf-parse yields fragmented or out-of-order token sequences. This affects TF-IDF vector quality and reduces scoring accuracy for non-standard documents. Addressing this through layout-aware PDF processing — for instance, integrating a LayoutLM-based extraction step — represents the most impactful near-term improvement. A second limitation is the static nature of TF-IDF weights: as job description vocabulary evolves across industries, periodic recalibration of the corpus is required to maintain scoring relevance [6].

## IX. CONCLUSION AND FUTURE WORK

This paper presented NextStep, an intelligent full-stack job portal built on the MERN stack with a real NLP resume scoring engine using TF-IDF vectorization and Cosine Similarity. The platform delivers a decoupled three-tier architecture with JWT-secured RBAC, MongoDB Atlas cloud persistence, and cloud-native deployment on Vercel and Render.com. Performance benchmarking confirmed API response times of approximately 150 ms, NLP analysis within 1–2 seconds, and frontend load times of approximately 1.2 seconds — all within production-grade thresholds. Security testing confirmed robust JWT validation and role enforcement across all endpoints.

Three development directions are prioritized for future work. In the near term, the resume parser will be enhanced with spaCy NER and layout-aware PDF processing to improve accuracy for graphically templated resumes. In the medium term, the TF-IDF engine will be supplemented with Word2Vec or BERT embeddings for deeper semantic understanding of resume-job relevance. Over the longer term, real-time WebSocket-based communication between recruiters and shortlisted candidates using Socket.io and an AI-powered mock interview preparation module will extend the platform toward a comprehensive intelligent recruitment ecosystem [15][16].

## REFERENCES

[1] S. Narayanan, K. Reddy, and T. Patel, "Revisiting keyword-based recruitment systems: Semantic gaps and their consequences," IEEE Trans. Human-Machine Systems, vol. 54, no. 3, pp. 289–300, Jun. 2024.

[2] A. Singh and R. Kaul, "Building full-stack applications with MERN stack: Architecture, REST API design, and performance considerations," Int. J. Comput. Sci. Eng., vol. 15, no. 2, pp. 78–89, 2024.

[3] M. Gupta and N. Joshi, "Role-based access control patterns in Node.js and Express.js REST architectures," in Proc. IEEE CONECCT, Bangalore, India, 2024, pp. 112–119.

[4] K. Reddy and T. Naidu, "JWT authentication and RBAC in decoupled React-Node.js recruitment platforms," in Proc. IEEE CCEM, Hyderabad, India, 2024, pp. 88–95.

[5] L. Wang, H. Chen, and J. Zhao, "ResumeParser-2024: Benchmarking NLP-based extraction from structured and unstructured resume formats," in Proc. ACL Findings, Bangkok, Thailand, 2024, pp. 341–352.

[6] Y. Zhang, M. Liu, and B. Sun, "Transformer-based resume understanding with layout-aware pre-training," IEEE Access, vol. 12, pp. 48210–48225, 2024.

[7] H. Li and S. Kim, "Collaborative filtering for cold-start job recommendation: A hybrid approach," J. Inf. Sci. Eng., vol. 39, no. 1, pp. 85–99, 2023.

[8] P. Mehta, A. Tiwari, and S. Rao, "Hybrid content-based and rule-based job matching for resource-constrained environments," Expert Syst. Appl., vol. 238, p. 121782, Mar. 2024.

[9] F. Qin, J. Xu, and L. Sun, "TF-IDF and Cosine Similarity for resume-job matching: Accuracy versus inference latency trade-offs," in Proc. IEEE BigData, Sorrento, Italy, 2023, pp. 1204–1211.

[10] S. Alam, R. Ahmed, and T. Khan, "Lightweight NLP scoring for job recommendation: Closing the gap with transformer models," in Proc. Int. Conf. Comput. Intell. (ICCI), Dhaka, Bangladesh, 2024, pp. 55–62.

[11] V. Choudhary and K. Patel, "JWT-based stateless authentication for scalable Node.js REST APIs," Proc. Int. Conf. Cyber Secur. Trust (CyberST), pp. 188–196, 2023.

[12] P. Raj, D. K. Sinha, and V. Jain, "Write performance and consistency in MongoDB Atlas under concurrent multi-user workloads," Int. J. Web Eng. Technol., vol. 19, no. 2, pp. 120–135, 2023.

[13] T. Allen and K. Morris, "Vercel and Render.com deployment for decoupled React-Node applications: Scalability analysis," in Proc. Int. Conf. Cloud Comput. (CloudCom), Rome, Italy, 2023, pp. 210–218.

[14] OWASP Foundation, "OWASP Top Ten Web Application Security Risks," 2023. [Online]. Available: https://owasp.org/Top10/

[15] R. Pinto, C. Lima, and F. Santos, "Measuring user satisfaction in AI-assisted job platforms: A mixed-methods evaluation," Int. J. Human-Computer Studies, vol. 181, p. 103147, Jan. 2024.

[16] S. Banerjee and A. Roy, "Real-time WebSocket communication in recruitment platforms using Socket.io," IEEE Trans. Serv. Comput., vol. 17, no. 1, pp. 340–353, Jan. 2024.