

AI Resume Analyzer: An NLP-Driven Full-Stack Platform for Intelligent Resume Screening and Candidate-Job Matching

Suhanee Sahoo

Student, Dept. of Computer Science & Engineering
GIFT Autonomous College, Bhubaneswar, Odisha, India

Tanmayee Satapathy

Student, Dept. of Computer Science & Engineering
GIFT Autonomous College, Bhubaneswar, Odisha, India

Asst. Prof. Bijayalaxmi Sahoo

Assistant Professor, Dept. of Computer Science & Engineering
GIFT Autonomous College, Bhubaneswar, Odisha, India

Abstract—In today's highly competitive job market, recruiters face an overwhelming volume of resumes that must be screened efficiently, while job seekers struggle to determine whether their resumes are tailored to specific job descriptions. This mismatch between candidate profiles and job requirements leads to inefficiencies on both sides of the hiring process. To address this challenge, the AI Resume Analyzer is proposed as a complete, web-based full-stack solution that intelligently analyses resumes against job descriptions using Natural Language Processing (NLP) techniques. The system provides candidates with actionable feedback, skill gap identification, keyword optimisation suggestions, and an overall match score, while also offering a dedicated recruiter panel for managing job postings and viewing ranked candidates, along with a super admin dashboard for overseeing the entire platform. The frontend is built using ReactJS with Vite for a fast and responsive interface, styled with Tailwind CSS, and uses React Router DOM for navigation among candidate, recruiter, and administrator screens, with Axios handling backend communication. The system implements JWT-based authentication with BCrypt password hashing to secure access across all user roles. Resume files are uploaded and stored using Multer and Cloudinary, ensuring reliable file delivery. The AI analysis engine uses pdf-parse for text extraction, TF-IDF vectorisation, and cosine similarity algorithms to compute a resume-to-job-description match score, providing candidates with precise, data-driven feedback. The platform adopts a three-tier MERN-stack architecture (MongoDB, Express.js, React, Node.js) and demonstrates how full-stack web technologies and NLP can be combined to produce an efficient, fair, and transparent recruitment tool.

Keywords—Resume screening; Natural Language Processing; TF-IDF; Cosine similarity; MERN stack; Applicant Tracking System; ReactJS; Node.js; MongoDB; Cloudinary; JWT authentication; Recruitment automation

I. INTRODUCTION

Modern recruitment processes are increasingly strained by the sheer volume of applications received for each open position. Recruiters routinely review hundreds of resumes for a single job posting, while job seekers struggle to understand whether their resume is aligned with the job description and what specific improvements would increase their chances of selection. Most existing platforms either offer simple resume uploads with no analytical feedback, or rely on rigid Applicant Tracking Systems (ATS) that automatically reject candidates on the basis of keyword mismatches without offering any path to improvement.

Traditional manual screening, although still common in small and medium-sized organisations, is inconsistent, time-consuming, and prone to unconscious bias [1]. At the same time, modern job portals such as LinkedIn, Naukri, and Indeed adopt a recruiter-centric design and offer little actionable feedback to candidates whose resumes are rejected. This asymmetry creates a feedback gap on the candidate side and an efficiency gap on the recruiter side.

The AI Resume Analyzer is motivated by the idea of creating a single platform that addresses both gaps simultaneously. The system uses Natural Language Processing (NLP) techniques—specifically PDF text extraction, TF-IDF vectorisation, and cosine similarity—to compute an objective resume-to-job-description match score and to generate a list of matched keywords, missing skills, and prioritised improvement suggestions. A triple-interface design (Candidate, Recruiter, and Super Admin) allows the same underlying analysis engine to serve all three user roles with role-appropriate views and permissions.

From a software engineering perspective, the project explores a three-tier MERN-stack architecture comprising ReactJS with Vite on the presentation tier, Node.js with Express.js on the application tier, and MongoDB Atlas on the data tier, with Cloudinary acting as an external content delivery network for resume file storage. JWT-based

authentication and BCrypt password hashing enforce role-based access control across all three user interfaces.

The remainder of this paper is organised as follows. Section II reviews related literature on resume screening, NLP-based parsing, and full-stack web technologies. Section III presents the system design and methodology. Section IV discusses implementation details. Section V presents the user interface design and results. Section VI describes the testing strategy. Section VII concludes with directions for future enhancements.

II. LITERATURE REVIEW

A. Traditional Resume Screening and ATS Limitations

Manual resume screening has been studied extensively as a source of inefficiency and inequity in recruitment. Sinha, Akhtar, and Kumar [1] surveyed NLP- and machine-learning-based approaches to resume screening and observed that manual review introduces “recruiter bias,” where personal preferences and unconscious stereotypes influence shortlisting decisions, and that different reviewers often apply inconsistent standards to similar resumes. Saatçi, Kaya, and Ünlü [2] further reported that ATS-based filtering frequently rejects strong candidates whose resumes lack specific keywords from the job description, even when the underlying experience is well aligned with the role.

A recurring observation across these studies is the absence of structured feedback for rejected candidates. Conventional ATS pipelines return a binary accept/reject decision without explaining which keywords or skills were missing, leaving candidates with no actionable path to improvement. The AI Resume Analyzer addresses this gap directly by producing a detailed report of matched keywords, missing skills, and prioritised suggestions in addition to a numeric match score.

B. NLP Techniques for Resume Parsing

Robust NLP techniques are central to any AI-driven resume screening system. The pdf-parse library has been studied widely as a reliable tool for extracting raw text from PDF resumes, converting structured document content into plain text suitable for downstream NLP processing. Term Frequency–Inverse Document Frequency (TF-IDF) vectorisation, introduced as part of classical information retrieval theory, is a well-established method for representing documents as weighted term vectors that emphasise rare, content-bearing words and de-emphasise common stop words.

Cosine similarity has been investigated as the primary scoring algorithm in recent resume-screening literature [2], [6]. It measures the angular similarity between the TF-IDF vector representations of a resume and a job description and produces a normalised score that can be interpreted as a percentage match. Keyword-extraction techniques further allow the system to identify domain-specific terms that are

present in the job description but absent from the resume, providing direct input to the feedback module.

More recent work has begun to explore Large Language Model (LLM)-based screening. Lo et al. [3] proposed a context-aware, explainable multi-agent framework using LLMs for resume screening, and Xu, Yamamoto, and Morishima [4] presented AutoScreen-FW, an LLM-based framework for automated screening. While LLM-based methods can capture richer semantic relationships than TF-IDF, they introduce significant computational, cost, and privacy overheads, motivating the use of lightweight statistical NLP in the present work.

C. Full-Stack Web Technologies (MERN Stack)

The choice of an appropriate technology stack is critical for building responsive, scalable web applications. The MERN stack—MongoDB, Express.js, React, and Node.js—has emerged as a popular choice for high-concurrency web applications because of its unified use of JavaScript across the entire stack and its strong open-source ecosystem.

MongoDB, as a NoSQL document database, offers schema flexibility that is well suited to storing heterogeneous resume structures and free-form job descriptions without rigid table definitions. React with Vite enables rich, single-page user interfaces with fast development feedback loops and supports dynamic rendering of analysis dashboards without full page reloads. Node.js, with its non-blocking I/O model, is particularly suitable for handling concurrent resume uploads and analysis requests, while Express.js provides a minimal, expressive framework for RESTful API design.

D. Authentication, Storage, and Third-Party Integration

Secure handling of user credentials and uploaded files is essential in any system that processes personal data such as resumes. JSON Web Tokens (JWT) provide a stateless authentication mechanism well suited to distributed web applications, while BCrypt remains a widely recommended choice for password hashing because of its adaptive cost factor. Together, these primitives enable role-based access control across multiple user interfaces without requiring server-side session storage.

For file storage, Cloudinary has been investigated as a reliable global Content Delivery Network (CDN) capable of storing and serving uploaded resume files and profile images with low latency, regardless of the user’s physical location. Multer, the standard Node.js middleware for handling multipart form data, integrates naturally with Cloudinary to process PDF resume uploads before they are persisted. Iconography libraries such as Lucide React provide a consistent visual language across the interface without introducing heavy design dependencies.

III. SYSTEM DESIGN AND METHODOLOGY

A. Proposed Solution

The AI Resume Analyzer is proposed as an end-to-end automated, AI-powered platform that addresses the inefficiency and feedback gaps of conventional manual and ATS-based resume screening. Candidates upload a PDF resume and a target job description and receive an immediate analysis report containing a match score, a list of matched keywords, a list of missing skills, and a prioritised set of improvement suggestions. Recruiters view ranked candidate lists for each of their job postings, sorted by AI-computed match score, while a super admin oversees the entire platform.

Unlike conventional ATS tools, which only filter resumes, the proposed system closes the feedback loop by returning the same analysis data that drives the recruiter's ranking back to the candidate. This dual use of a single analysis result—as both a recruiter ranking signal and a candidate self-improvement signal—is the core methodological contribution of the system.

B. Three-Tier MERN Architecture

The backend adopts a classical three-tier architecture based on the MERN stack:

- **Presentation Tier (Frontend):** Developed using ReactJS with Vite. Handles user interfaces for Candidates, Recruiters, and Super Admins, with React Router DOM for navigation and Axios for API calls.
- **Application Tier (Backend):** Built with Node.js and Express.js. Contains the business logic, RESTful API routes, the NLP analysis engine, and integration code for external services such as Cloudinary.
- **Data Tier (Database):** Uses MongoDB Atlas to store user credentials, resume metadata, job descriptions, and analysis reports as flexible JSON-like documents.

External services are integrated at the application tier. Cloudinary serves as the file storage and delivery layer for resumes and profile images, while the NLP analysis engine is implemented internally inside the Node.js process using the pdf-parse and natural libraries.

C. Selected Technologies

1) Frontend

ReactJS with Vite as the build tool provides a fast and responsive single-page interface. Tailwind CSS is used for utility-first styling and responsive design, Lucide React supplies consistent iconography, React Router DOM handles client-side routing across the three user role interfaces, and Axios manages HTTP communication with the backend. Context API is used for lightweight application state management.

2) Backend

Node.js (v18 or higher) acts as the runtime environment, with Express.js exposing RESTful APIs over HTTP/HTTPS.

Multer is used as middleware for multipart file uploads, and JWT-based middleware enforces authentication and role-based access control on protected routes. Server-side input validation is implemented using express-validator.

3) Database and Storage

MongoDB Atlas hosts the application database in the cloud, with five primary collections: Users, Resumes, JobDescriptions, AnalysisReports, and Admins. Cloudinary stores uploaded PDF resume files and profile images, returning persistent URLs and public IDs that are saved in the corresponding MongoDB documents.

4) Security

Security is implemented in depth across multiple layers. BCrypt is used for password hashing prior to storage, ensuring that plain-text credentials are never persisted. JWTs are stored in HTTP-only cookies to prevent client-side JavaScript access and mitigate XSS-based token theft. CORS is configured at the backend to restrict API access to registered frontend origins, and role-based middleware functions guard recruiter and admin endpoints.

5) AI Analysis Engine

The internal NLP analysis engine uses the pdf-parse library to extract raw text from uploaded PDF resumes, the natural library for TF-IDF vectorisation, and a cosine similarity computation to produce a percentage match score. Keyword extraction over the same TF-IDF representation identifies missing skills and matched terms, which are then transformed into human-readable improvement suggestions.

D. System Workflow

The end-to-end workflow proceeds as follows. A candidate first authenticates against the backend, which issues a signed JWT and stores it in an HTTP-only cookie. The candidate then uploads a PDF resume and enters a target job description via the frontend. Multer receives the multipart upload at the backend, forwards the file to Cloudinary for storage, and persists the returned URL and public ID in the Resumes collection.

The backend then invokes pdf-parse on the file buffer to extract raw text, which is sanitised using regular expressions to remove non-printable characters and normalise whitespace. The cleaned resume text and the job description are converted into TF-IDF vectors and compared using cosine similarity to produce a raw similarity score. The score is normalised onto a 0–100 scale, the missing and matched keyword sets are computed, and a set of improvement suggestions is generated. The full report is persisted to the AnalysisReports collection and returned to the frontend, where it is rendered on the candidate dashboard. Recruiter dashboards aggregate AnalysisReports keyed by job posting and display candidates ranked in descending order of match score.

E. Database Architecture

The Users collection stores identity data for candidates, recruiters, and admins, including name, email, BCrypt-hashed password, role field, and a Cloudinary URL for the profile image. The Resumes collection holds the Cloudinary file URL, Cloudinary public ID, raw extracted text from pdf-parse, upload timestamp, and a reference to the owning candidate. The JobDescriptions collection holds all job postings created by recruiters, including job title, company name, required skills, full description text, and a reference to the recruiter.

The AnalysisReports collection acts as the junction entity that links a specific resume with a specific job description. It stores the computed match score, the array of matched keywords, the array of missing skills, the generated improvement suggestions, and the analysis timestamp. A separate Admins collection holds super-admin credentials for the platform-wide dashboard. The dominant cardinality relationships are 1 : M between Users and Resumes, 1 : M between recruiters and JobDescriptions, and M : N between Resumes and JobDescriptions through AnalysisReports.

IV. IMPLEMENTATION

A. Authentication Module (JWT and BCrypt)

User passwords are hashed using BCrypt before being stored in the Users collection, ensuring that plain-text credentials are never persisted. Upon successful login, the backend issues a signed JWT containing the user's ID and role and stores it in an HTTP-only cookie to prevent client-side script access. JWT-verification middleware decodes and validates the token on every protected route, and additional role-checking middleware functions (requireAuth, requireRecruiterRole, requireAdminRole) restrict access to recruiter and admin endpoints based on the role claim. The frontend mirrors this behaviour with route guards that redirect unauthenticated users to the login page.

B. Resume Upload and Parsing Module

Multer is configured as backend middleware to accept multipart form data, with a fileFilter that rejects all non-PDF uploads and a file-size limit that prevents excessively large files from consuming server resources. Accepted files are streamed to Cloudinary, which returns a persistent URL and a public ID that are stored in the Resumes collection along with the upload date and the owning candidate's reference.

The pdf-parse library is then invoked on the file buffer to extract raw text. Because real-world resumes often use multi-column layouts, tables, and embedded graphics, the raw output can contain garbled word order and missing whitespace. To mitigate this, a sanitisation pipeline normalises whitespace, removes non-printable characters, and corrects common extraction artefacts before the text is passed to the analysis engine.

C. AI Analysis and Scoring Module

The resume text and job description text are processed using TF-IDF vectorisation provided by the natural NLP library. A cosine similarity comparison between the two TF-IDF vectors yields a raw similarity score. During development it was observed that the raw scores tended to cluster at either very low or very high values, making it difficult for candidates to differentiate good resumes from excellent ones. A normalisation formula is therefore applied to redistribute scores more meaningfully across the 0–100 range so that the displayed percentage better reflects the practical quality of the match.

In parallel with the score computation, the system performs keyword gap analysis. Significant terms in the job description are extracted using the same TF-IDF representation and compared against terms present in the resume text. Terms present in both sets are reported as matched keywords, while terms present only in the job description are reported as missing skills. The missing-skills list is then transformed into human-readable improvement suggestions using a template-based generator.

D. Job Description Matching and Ranking Module

Recruiters create detailed job descriptions through a dedicated frontend interface; each description is stored in the JobDescriptions collection along with required skills, experience level, and qualifications, and is referenced by all subsequent analyses against that posting. When a candidate submits an analysis against a specific job description, the resulting match score is persisted in the AnalysisReports collection. The recruiter dashboard then computes a ranked leaderboard for each posting by querying AnalysisReports filtered on the job ID and sorted by match score in descending order. Backend validation prevents the same candidate from analysing the same resume against the same job description more than once, preserving the integrity of the leaderboard.

E. Dashboard and Feedback Module

The candidate dashboard displays the user's analysis history, their best overall score, and quick links to upload a new resume or open a specific report. Each detailed report view renders the match score as an animated circular progress gauge, lists matched keywords in green badges and missing skills in red badges, and shows the prioritised improvement suggestions as a bullet list. A download button allows the candidate to export the report as a PDF for offline review.

The recruiter dashboard summarises active job postings, total analyses received, and the top-scoring candidates per posting, and provides a dynamic ranked candidate table for each job description. The super-admin dashboard provides global statistics such as total registered candidates and recruiters, total resumes uploaded, total analyses performed, and content-moderation controls. Real-time consistency across these dashboards is achieved using React's useEffect

hooks with dependency arrays that trigger data refetching whenever a new analysis is submitted, eliminating the need for manual page refreshes.

V. USER INTERFACE DESIGN AND RESULTS

A. User Interface Design

The candidate frontend consists of a landing page with a clear value proposition, a dedicated upload portal that accepts drag-and-drop PDF uploads and a job description text area, an analysis results view that prominently displays the match score gauge and the side-by-side keyword comparison, and an analysis history panel for revisiting past reports. A consistent, fully responsive navigation bar adapts the layout to mobile, tablet, and desktop screens.

The recruiter admin panel includes a personal overview dashboard, a job description manager with a rich text editor for entering detailed requirements, and a dynamic ranking table for each job posting. The super admin dashboard adds global analytics, a consolidated user-management view that allows the administrator to activate or deactivate accounts, content moderation tools for removing inappropriate postings or resumes, and a system health monitor reporting API response times and database storage utilisation.

B. System Implementation Results

The authentication module delivers fully functional JWT-based authentication across all three user roles, with BCrypt-hashed passwords and HTTP-only cookie storage. The resume upload and parsing pipeline correctly handles PDF uploads of varied formatting, with the sanitisation step significantly improving extracted-text quality for multi-column and table-heavy resumes.

The NLP scoring engine produces meaningful and differentiated match scores across diverse resume–job description combinations after score normalisation, and the keyword gap analysis consistently identifies relevant missing technical skills. Cloudinary delivers uploaded files reliably through its global CDN, and MongoDB Atlas provides low-latency access to user and analysis data.

Compared with manual screening, the system reduces per-resume analysis time from several minutes of human reading to under ten seconds of automated processing. Because the same TF-IDF-based scoring rule is applied to every resume, the system also delivers more consistent evaluation than human reviewers, and—unlike conventional ATS rejection—every candidate receives a detailed feedback report.

C. Discussion

Three integration challenges shaped the final design. First, PDF parsing accuracy varied substantially with resume layout, which motivated the addition of the regex-based text sanitisation pipeline before TF-IDF processing. Second, raw cosine similarity values tended to cluster at extreme ends of

the 0–1 range, which motivated the score-normalisation formula that redistributes values more meaningfully across the displayed 0–100 scale. Third, real-time data consistency across the three role-specific dashboards required careful use of useEffect dependency arrays so that new analyses propagate to the recruiter ranking without manual refreshes.

Together, these solutions ensure that the AI Resume Analyzer is robust, user-friendly, and technically sound. The architecture also generalises naturally to future enhancements such as LLM-based semantic scoring [3], [4], video interview integration, blockchain credential verification, and native mobile clients.

D. Limitations

The primary limitation of the current system is the use of statistical TF-IDF and cosine similarity rather than deeper semantic representations. TF-IDF treats terms independently and does not capture synonyms or contextual relationships between skills, which can occasionally penalise resumes that use different vocabulary to describe the same competence. Integration of pretrained language-model embeddings or LLM-based semantic scoring [3], [4] is identified as the highest-impact future improvement. Additionally, the platform currently operates with a fixed set of resume templates, depends on continuous internet connectivity, and supports only a single language. PDF customisation options for the downloadable report are also currently limited.

VI. TESTING

Testing was conducted across three levels—unit testing, integration testing, and system testing—using Thunder Client and Postman for backend API testing alongside manual UI verification.

Unit testing covered individual backend functions such as the score-calculation utility and the keyword-extraction helper, individual React components such as the ScoreGauge and KeywordBadge, and client-side form validation rules that prevent incomplete or malformed upload requests. Integration testing verified end-to-end communication between the frontend Axios calls and backend endpoints, the JWT issuance and validation flow across login and protected routes, the file upload pipeline from the React form through Multer to Cloudinary, and the pdf-parse extraction step feeding the NLP analysis engine.

System testing simulated full user journeys: candidate registration and login, resume upload, analysis submission, score display, and re-visiting historical reports; recruiter login, job posting creation, and review of the ranked candidate table; and super-admin oversight of users and content. Role-based access tests confirmed that candidates cannot reach recruiter or admin routes, and that unauthenticated users are redirected to the login page on access to any protected route. Responsiveness was verified across mobile, tablet, and desktop viewports, and duplicate-

analysis prevention was confirmed by attempting to submit the same resume against the same job description twice.

VII. CONCLUSION AND FUTURE WORK

This paper has presented the AI Resume Analyzer, a full-stack, NLP-driven platform for intelligent resume screening and candidate-job matching. By combining a three-tier MERN architecture with PDF text extraction, TF-IDF vectorisation, and cosine similarity, the system replaces inefficient manual screening with an objective, data-driven workflow that benefits candidates, recruiters, and platform administrators simultaneously.

Three distinct, secure interfaces have been delivered: a Candidate Portal that provides resume upload, instant AI analysis, and actionable feedback; a Recruiter Admin Panel that offers job posting management and an automated candidate ranking system; and a Super Admin Dashboard that provides comprehensive platform oversight. Role-based JWT authentication, BCrypt password hashing, HTTP-only cookies, and CORS-restricted APIs together secure the platform against common web-application threats. Cloudinary provides reliable global file delivery, and pdf-parse with a custom sanitisation pipeline ensures robust text extraction across varied resume formats.

Several enhancements are identified as high-value future work. The most impactful would be integration of LLM-based semantic scoring [3], [4] to capture synonyms and contextual relationships that statistical TF-IDF cannot represent, supplemented by an automatic resume-rewriting feature that suggests specific bullet-point rewrites for missing skills. Video interview integration with WebRTC, automated interview scheduling from the recruiter dashboard, blockchain-based credential verification for tamper-proof skill badges, and native React Native mobile clients with camera-based resume scanning and offline access are also identified as natural next steps. On the security side, two-factor authentication for recruiter and admin accounts, AES-256 encryption at rest for uploaded files, and API rate limiting with DDoS protection would further strengthen the platform for production deployment.

REFERENCES

- [1] A. Sinha, M. A. K. Akhtar, and A. Kumar, "Resume screening using natural language processing and machine learning: A systematic review," in *Machine Learning and Information Processing*, Singapore: Springer, 2021, doi: 10.1007/978-981-33-4859-2_21.
- [2] M. Saatçı, R. Kaya, and R. Ünlü, "Resume screening with Natural Language Processing (NLP)," *Alphanumeric Journal*, vol. 12, no. 2, 2024.
- [3] F. P.-W. Lo, J. Qiu, Z. Wang, H. Yu, Y. Chen, G. Zhang, and B. Lo, "AI hiring with LLMs: A context-aware and explainable multi-agent framework for resume screening," *arXiv preprint arXiv:2504.02870*, 2025.
- [4] Z. Xu, S. Yamamoto, and A. Morishima, "AutoScreen-FW: An LLM-based framework for resume screening," *arXiv preprint arXiv:2603.18390*, 2026.
- [5] P. Parasurama and J. Sedoc, "Degendering resumes for fair algorithmic resume screening," *arXiv preprint arXiv:2112.08910*, 2021.
- [6] "Automated resume screening system using NLP and machine learning," *ResearchGate Publication*, 2025.
- [7] "AI-powered automated resume screening and job matching system using NLP and machine learning," *ResearchGate Publication*, 2025.
- [8] Todupunuri, A. (2024). Exploring the use of generative AI in creating deepfake content and the risks it poses to data integrity, digital identities, and security systems. Available at SSRN 5014688.
- [9] "Improving resume screening with NLP and machine learning: Addressing efficiency and fairness," *Grenze International Journal of Engineering and Technology*, 2025.
- [10] "An intelligent resume screening and job matching platform using NLP and machine learning," *International Journal of Engineering Research & Technology (IJERT)*, 2026.
- [11] Poojari, R. Frameworks for Data Management and Lineage in Large-Scale Healthcare Data Systems.
- [12] OpenJS Foundation, "Node.js documentation," 2024. [Online]. Available: <https://nodejs.org/docs/>
- [13] Purmani, S. S. R. (2025). Streamlining IT operations and service management with agile frameworks. *European Journal of Advances in Engineering and Technology*, 12(4), 76–81.
- [14] Cloudinary, "Cloudinary developer documentation," 2024. [Online]. Available: <https://cloudinary.com/documentation>
- [15] OWASP Foundation, "OWASP Top Ten Project," 2024. [Online]. Available: <https://owasp.org/www-project-top-ten/>
- [16] Mudusu, S. K. (2026, February 9). AI-augmented data quality engineering. *InfoWorld (Foundry Expert Contributor Network)*.
- [17] Maturi, S. Y. (2022). Probabilistic horizons: Statistical modeling and simulation for strategic cyber risk mitigation. *Journal of Information Systems Engineering and Management*, 7(2).
- [18] Ravishankara, M. (2026, February). PlotChain: Deterministic Checkpointed Evaluation of Multimodal LLMs on Engineering Plot Reading. In *SoutheastCon 2026* (pp. 1-8). IEEE.
- [19] Venkata Ramana, P. (2024). AI-driven predictive analytics in ERP systems for proactive supply chain optimization. *International Journal of Research in Information Technology and Computing*, 8(4).