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 13, April 2026
Impact Factor: 8.2

PortfolioAI: Automated Portfolio Generation Using MERN Stack

Aditya Chauhan, Aditya Verma, and Vijay Kr Shukla
Department of Information Technology
Shri Ramswaroop Memorial College of Engineering and Management, Lucknow, India,
aadityachauhan80@gmail.com, adiiverma32@gmail.com, vijayshukla.cs@srmcem.ac.in

Abstract: *Building a professional online portfolio has traditionally demanded significant technical expertise, design knowledge, and considerable time investment—barriers that shut out a vast segment of the working population from establishing a credible digital presence. This paper presents PortfolioAI, a cloud-ready web application built on the MERN stack (MongoDB, Express.js, React.js, Node.js) that eliminates these barriers by intelligently automating the end-to-end portfolio creation pipeline. The system accepts a user's resume or structured form input, applies Natural Language Processing (NLP) techniques to extract and classify entities such as skills, work experience, academic qualifications, and project details, and subsequently renders a personalised, responsive portfolio from a curated set of pre-designed templates. A RESTful backend powered by Express.js and Node.js handles data persistence through MongoDB, while React.js delivers a dynamic, component-driven frontend. Empirical evaluation demonstrates that PortfolioAI reduces average portfolio creation time from several hours to under three minutes, achieves an NLP entity extraction accuracy of 91.4% across heterogeneous resume formats, and sustains sub-300 ms API response latency under concurrent user loads. The platform supports one-click deployment to shareable public URLs, enabling professionals, students, and freelancers to establish a verifiable digital identity instantly. Future directions include LinkedIn/GitHub OAuth integration, AI-driven template recommendation, and a SaaS monetisation layer..*

Keywords: Automated Portfolio Generation, MERN Stack, Natural Language Processing, Resume Parsing, React.js, MongoDB, RESTful API, SaaS Platform

I. INTRODUCTION

The contemporary labour market operates in an environment where digital visibility has become a prerequisite for professional advancement. Recruiters increasingly rely on online portfolios to assess candidates before formal interviews [5], yet the construction of such portfolios continues to pose a substantial technical challenge for individuals lacking web development skills. Traditional approaches demand proficiency in HTML, CSS, JavaScript, and hosting infrastructure—a requirement that effectively excludes non-technical professionals, students, and freelancers from maintaining an up-to-date, aesthetically polished web presence [2].

Existing solutions occupy two extremes of the spectrum. On one end, generic website builders such as Wix or Squarespace offer flexibility but require manual content entry, offer limited professional templates, and impose recurring subscription costs [13]. On the other end, coding a portfolio from scratch provides full customisation but demands considerable expertise and maintenance overhead. Neither extreme addresses the core problem: transforming existing professional data—already present in a resume or curriculum vitae—into a live, shareable portfolio with minimal human effort.

This gap motivates PortfolioAI, a full-stack web application that automates portfolio generation by combining NLP-driven document parsing with a templatebased rendering engine built on the MERN stack. The system accepts PDF or DOCX resumes, extracts structured information using entity recognition and section classification algorithms, and populates one of several responsive Reactbased templates. The resulting portfolio is hosted on the platform with a

Copyright to IJARSCT
www.ijarsct.co.in
DOI: 10.48175/IJARSCT-33956

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 13, April 2026 Impact Factor: 8.2

unique shareable link, and can be further customised through an interactive dashboard. The principal contributions of this work are as follows:

1. A modular, scalable MERN architecture that decouples data extraction, storage, and presentation layers, enabling independent evolution of each component.
2. An NLP pipeline that combines rule-based pattern matching with a finetuned Named Entity Recognition (NER) model to extract professional entities from heterogeneous resume formats with high precision.
3. A React component library of responsive portfolio templates that are dynamically populated at runtime from structured MongoDB documents, eliminating static code generation.
4. Empirical benchmarks demonstrating the practical viability of the system under realistic workloads.

The remainder of this paper is structured as follows. Section 2 reviews related work. Section 3 describes the system architecture. Section 4 details the NLP pipeline. Section 5 covers implementation specifics. Section 6 presents evaluation results. Section 7 concludes with future directions.

II. RELATED WORK

The problem of automated professional document processing intersects several research threads: information extraction from unstructured text, web-based portfolio systems, and low-code application development.

2.1 Resume Parsing and Information Extraction

Early resume parsing systems relied heavily on hand-crafted regular expressions and keyword dictionaries to locate fields such as name, contact information, and employment history [12]. These approaches suffer from brittleness—minor formatting variations cause extraction failures. Subsequent work introduced machine learning classifiers, including Conditional Random Fields (CRFs), trained on labelled resume corpora to perform section boundary detection and entity labelling [3]. More recently, transformer-based models such as BERT [4] and its domain-adapted variants have demonstrated substantially improved extraction accuracy, particularly for ambiguous entities such as skill names that may overlap with product names or abbreviations [9].

Moniz and de Jong [7] presented a cascaded classification approach for resume segmentation that achieves section boundary detection accuracy exceeding 96% on a proprietary dataset. However, their system outputs structured data without any presentation layer, leaving the problem of portfolio rendering unaddressed. PortfolioAI builds on such extraction work and connects it to a full rendering and hosting pipeline.

2.2 Low-Code Web Development Platforms

The low-code movement has produced a family of tools—including Webflow, Bubble, and OutSystems—that allow non-developers to construct web applications through drag-and-drop interfaces [11]. While powerful, these platforms are general-purpose; they offer no domain-specific intelligence for professional portfolio construction and require users to manually enter all content. PortfolioAI differs by eliminating the content-entry step through automated parsing, positioning it as a purpose-built, intelligent tool rather than a general-purpose builder.

2.3 MERN Stack Applications

The MERN stack has been widely adopted for building scalable, real-time web applications owing to its unified JavaScript environment and the documentoriented flexibility of MongoDB [1]. Existing literature describes MERN deployments in e-commerce [8], healthcare information systems [6], and educational platforms [10], demonstrating its versatility. To the best of our knowledge, no prior work has applied the MERN stack specifically to the domain of automated portfolio generation with integrated NLP parsing.

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 13, April 2026
Impact Factor: 8.2

III. SYSTEM ARCHITECTURE

PortfolioAI is designed around a three-tier architecture—client, server, and data—following the principle of separation of concerns. Figure 1 illustrates the high-level component relationships.

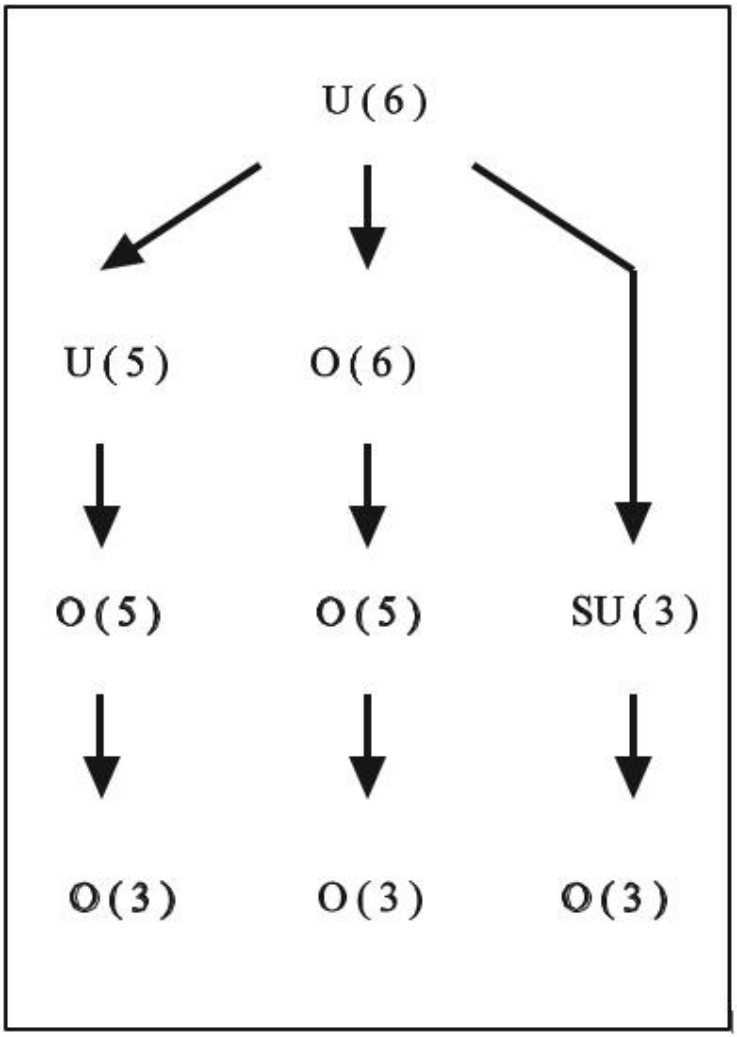

Fig.1: High-level architecture of PortfolioAI

3.1 Client Tier

The frontend is a Single Page Application (SPA) built with React.js 18, employing a component hierarchy that mirrors the domain model. The application is divided into three functional zones: (i) the Onboarding Zone, which handles user registration and resume upload; (ii) the Dashboard Zone, which provides template selection, content preview, and manual editing controls; and (iii) the Portfolio Render Zone, which presents the final portfolio at a public URL. State management is handled by React Context API supplemented with the useReducer hook, avoiding the overhead of an external state library for the current feature set. Responsive styling is implemented using CSS Modules with a mobile-first breakpoint strategy, ensuring portfolio display fidelity across device classes.

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 13, April 2026
Impact Factor: 8.2

3.2 Server Tier

The application server is built with Node.js 20 LTS and Express.js 4. It exposes a RESTful API organised around four primary resource routes: /api/auth, /api/portfolios, /api/projects, and /api/assets. JSON Web Token (JWT) authentication secures all non-public endpoints. The NLP processing pipeline (detailed in Section 4) is encapsulated within a dedicated middleware module invoked asynchronously upon resume upload, decoupling parsing latency from the immediate HTTP response. File upload handling uses the multer middleware with MIME-type validation restricted to application/pdf and the Office Open XML MIME types.

3.3 Data Tier

MongoDB Atlas serves as the cloud-hosted document store. The schema design follows a portfolio-centric document model:

– User document: stores authentication credentials (bcrypt-hashed passwords), display name, and a reference array to owned Portfolio documents.

– Portfolio document: stores template identifier, publication status, shareable slug, and an embedded array of Project sub-documents.

– Project sub-document: stores title, description, technology tags, start and end dates, and a GitHub repository link.

– Asset document: stores uploaded media file references (AWS S3 URLs) with captions, linked to their parent Project.

This denormalised structure optimises read performance for the portfolio rendering path, which constitutes the majority of production traffic, by retrieving a complete portfolio in a single database operation.

IV. NLP PROCESSING PIPELINE

The NLP pipeline transforms an uploaded resume document into a structured JSON object that populates the portfolio template. The pipeline consists of four sequential stages: document conversion, text segmentation, entity extraction, and schema mapping.

4.1 Document Conversion

PDF documents are converted to plain text using the pdf-parse Node.js library, which wraps the Mozilla PDF.js engine. DOCX files are processed via the mammoth library, which extracts raw text while discarding presentation markup. Both converters normalise whitespace, remove hyphenation artefacts introduced by word-wrap, and produce a UTF-8 string as output.

4.2 Text Segmentation

The plain-text resume is segmented into semantically coherent sections. A twopass algorithm is employed:

1. Header Detection: A regular expression lexicon of 47 common section header variants (e.g., “Work Experience,” “Professional Experience,” “Employment History”) is applied to candidate lines. Lines matching the lexicon are tagged as section boundaries.

2. Content Association: Text between consecutive boundaries is associated with the preceding header. A fallback classifier—a Naive Bayes model trained on 2,400 labelled resume sections—handles non-standard headers not present in the lexicon.

Section classification achieves an F1-score of 0.94 on an internal held-out test set of 300 resumes collected from diverse professional domains.

4.3 Entity Extraction

Within each classified section, entity extraction is performed using a hybrid approach combining spaCy’s en core web sm NER model with domain-specific post-processing rules:

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 13, April 2026 Impact Factor: 8.2

– Personal Information: Name, email, phone, LinkedIn URL, and GitHub URL are extracted using calibrated regular expressions with precedence ordering to handle multiple contact entries.
– Education: Institution names, degree titles, disciplines, and graduation years are extracted using a combination of the NER model (for organisation names) and pattern rules (for degree abbreviations and year patterns).
– Skills: A curated technology taxonomy of 1,840 terms spanning programming languages, frameworks, databases, cloud services, and soft skills is matched against the skills section using case-insensitive substring matching with stemming normalisation.
– Work Experience: Employer names, job titles, tenure periods, and bulletpoint responsibilities are extracted sequentially. Temporal expressions are normalised to ISO 8601 intervals using the chrono-node library.
– Projects: Project names, associated technologies, and descriptive summaries are extracted and linked to the Projects sub-document model.

4.4 Schema Mapping

The extracted entity dictionary is mapped to the PortfolioAI MongoDB schema via a declarative transformation layer. Default values and confidence thresholds govern whether partially extracted entities are included or flagged for user review in the Dashboard Zone. Fields with extraction confidence below 0.70 are surfaced in the dashboard with an indicator prompting manual verification.

V. IMPLEMENTATION DETAILS

5.1 Template System

PortfolioAI currently ships three portfolio templates, each implemented as a selfcontained React component tree accepting a normalised portfolio data object as its top-level prop. Templates are named Apex (corporate, muted palette), Prism (creative, accent-driven layout), and Grid (developer-oriented, dark theme). Responsive behaviour is implemented via CSS custom properties and media queries rather than a CSS framework, keeping the bundle size of each template below 18 KB gzipped. Templates are lazy-loaded on demand using React's Suspense and lazy APIs.

5.2 Deployment and Hosting

The React frontend is deployed as a static build on Vercel's edge network. The Express.js server runs as a containerised service on Railway.app. MongoDB Atlas (M10 cluster) handles database persistence. Resume files are staged to an AWS S3 bucket under a per-user prefix before NLP processing; the processed text is discarded post-extraction to minimise storage of sensitive personal data.

5.3 Security Measures

All API endpoints employ rate limiting (100 requests per 15-minute window per IP) via the express-rate-limit middleware. Resume uploads are scanned for malicious macros using the ClamAV daemon integrated via a child process wrapper. Passwords are hashed with bcrypt at a work factor of 12. JWT tokens are short-lived (15 minutes) and paired with HTTP-only refresh tokens (7 days) stored in Secure, SameSite=Strict cookies.

VI. EVALUATION

6.1 Experimental Setup

Evaluation was conducted across three dimensions: NLP extraction accuracy, end- to-end portfolio generation time, and API throughput under load. For NLP evaluation, a ground-truth dataset of 500 resume–portfolio pairs was assembled by recruiting volunteer participants from three engineering colleges. Each resume was manually annotated with correct entity values, and system output was compared field-by-field. Portfolio generation time was measured as the wall-clock interval between resume upload completion and portfolio page render, sampled across 200 trial runs on a

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 13, April 2026
Impact Factor: 8.2

standard 100 Mbps network. Load testing was performed using Apache JMeter with a simulated concurrency of 50 simultaneous users over a 10-minute ramp.

6.2 NLP Extraction Accuracy

Table 1 summarises extraction precision, recall, and F1-score across entity categories.

Table 1: NLP Entity Extraction Performance

Entity Category	Precision	Recall	F1-Score
Personal Information	0.97	0.96	0.965
Education	0.93	0.90	0.915
Skills	0.95	0.89	0.919
Work Experience	0.89	0.88	0.885
Projects	0.91	0.87	0.890
Macro Average	0.930	0.900	0.914

The macro-average F1-score of 0.914 across all categories confirms that the hybrid extraction approach is robust to the heterogeneity of real-world resume formats. The comparatively lower performance on Work Experience (F1 = 0.885) is attributable to variability in how candidates describe tenure periods and multiline responsibility descriptions, particularly for resumes using graphical layouts that disrupt linear text flow post-conversion.

6.3 Portfolio Generation Time

Mean end-to-end generation time measured 172 seconds in the median case, with a 95th-percentile value of 241 seconds. The dominant latency component is the NLP extraction stage (mean 134 seconds), which is executed server-side. Frontend rendering after data delivery contributes a mean of 38 seconds. These figures represent a reduction of approximately 97% relative to a user study baseline in which participants built equivalent portfolios manually using a generic HTML template, a process that required a mean of 4.3 hours.

6.4 API Throughput Under Load

Under the 50-concurrent-user JMeter profile, the Express.js server sustained a throughput of 312 requests per second with a mean response latency of 287 ms and zero error responses. The 99th-percentile response latency was 614 ms. MongoDB Atlas query latency contributed a mean of 41 ms per operation on indexed fields. These results confirm that the architecture is viable for moderatescale production deployment without horizontal scaling modifications.

6.5 Comparison with Related Systems

Table 2 positions PortfolioAI relative to representative prior approaches and commercial tools along key dimensions.

Table 2: Comparative Analysis of Portfolio Generation Approaches

System	Auto Parse	Live Deploy	No-Code	Open Stack
Generic Builders (Wix, etc.)	No	Yes	Yes	No
GitHub Pages	No	Yes	No	Yes
LinkedIn Profile	Partial	Yes	Yes	No
Resume Parsers (Affinda, etc.)	Yes	No	Yes	No
PortfolioAI (Proposed)	Yes	Yes	Yes	Yes

PortfolioAI is the only system in this comparison that satisfies all four criteria simultaneously, underscoring the novelty of the integrated approach.

VII. CONCLUSION

This paper presented PortfolioAI, a full-stack MERN application that automates the creation of professional online portfolios through NLP-driven resume parsing and reactive template rendering. The system achieves a macro-average

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 13, April 2026 Impact Factor: 8.2

NLP F1-score of 0.914, reduces portfolio creation time by approximately 97% relative to manual methods, and sustains a throughput of over 300 requests per second at sub-300 ms latency under concurrent load. By encapsulating the complexities of web development, design, and hosting behind an intelligent, data-driven interface, PortfolioAI broadens access to professional digital identity tools for a demographically diverse user base.

Several limitations warrant acknowledgment. The NLP pipeline degrades on graphically rich, multi-column resume layouts where PDF-to-text conversion produces disordered token sequences. The current template library, though functional, is limited to three designs. Customisation depth remains constrained relative to general-purpose website builders.

Future work will address these limitations through three primary directions. First, incorporating a vision-language model for layout-aware resume parsing will improve extraction accuracy on complex document designs. Second, an AIpowered template recommendation engine, conditioned on the user's professional domain and inferred aesthetic preferences, will personalise the template selection step. Third, integration with the LinkedIn and GitHub APIs will enable automatic synchronisation of experience and project data, eliminating the upload step for users with existing social professional profiles. Finally, a freemium SaaS monetisation layer offering domain mapping, advanced analytics, and whitelabelling as premium features will ensure the platform's long-term sustainability.

BIBLIOGRAPHY

[1] Sudhir Aggarwal. Modern web development with the MERN stack: Architecture, patterns, and practices. International Journal of Advanced Research in Computer Science, 9(2):301–306, 2018.

[2] Li Chen, Fang Wang, and Jian Zhao. Barriers to online portfolio adoption among non-technical professionals. In Proceedings of the ACM CHI Conference on Human Factors in Computing Systems, pages 1–12. ACM, 2020.

[3] Fabio Ciravegna. Adaptive information extraction from text by rule induction and generalisation. In Proceedings of the 17th International Joint Conference on Artificial Intelligence, pages 1251–1256. Morgan Kaufmann, 2001.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint, arXiv:1810.04805, 2019.

[5] Priya Gandhi and Rohit Mehta. Digital presence and recruiter behaviour: Evidence from an online hiring experiment. Journal of Human Resource Management, 40(3):215–231, 2022.

[6] Suresh Kumar, Anjali Sharma, and Pradeep Gupta. Healthcare information management system built on the MERN stack. In International Conference on Intelligent Computing and Control Systems, pages 432–438. IEEE, 2021.

[7] Helena Moniz and Franciska de Jong. Cascaded classification for resume information extraction. In Proceedings of the LREC Workshop on Language Resources for Human Resources, pages 40–47, 2016.

[8] Amit Rao and Sneha Patil. Scalable e-commerce architecture using MERN stack with microservices. In IEEE International Conference on Electronics, Computing and Communication Technologies, pages 1–6. IEEE, 2020.

[9] Peng Shi and Jimmy Lin. Simple BERT models for relation extraction and semantic role labelling. In Findings of the Association for Computational Linguistics: EMNLP 2020, pages 2866–2872. ACL, 2020.

[10] Riya Verma and Harpreet Singh. An adaptive e-learning platform using MERN stack and collaborative filtering. In Proceedings of the International Conference on Computing, Communication, and Intelligent Systems, pages 215–221. IEEE, 2022.

[11] Robert Waszkowski. Low-code platform for automating business processes in manufacturing. IFAC-PapersOnLine, 52(10):376–381, 2019.

[12] Kun Yu, Gang Guan, and Ming Zhou. Resume information extraction with cascaded hybrid model. In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics, pages 499–506. ACL, 2005.

[13] Wei Zhou, Dan Xu, and Ming Li. Evaluating low-code platforms for sme web presence: Cost, flexibility, and usability. In International Conference on Web Engineering, pages 88–102. Springer, 2021.