



INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X

Impact Factor: 6.078

(Volume 12, Issue 3 - V12I3-1151)

Available online at: <https://www.ijariit.com>

CentralResume: A Unified Protocol for Structured Resume Management and Interoperable Job Profile Sharing

Anish Araz

anisharaz919@gmail.com

Jain University, Karnataka

Niraj Shah Rauniyar

shahniraj440@gmail.com

Jain University, Karnataka

Piyush Raj

piyushxcoder@gmail.com

Jain University, Karnataka

Dr. Gowthul Alam M M

gowthul.alam@jainuniversity.ac.in

Jain University, Karnataka

ABSTRACT

Creating and maintaining job profiles across multiple recruitment platforms is a redundant process. Users are often required to manually enter the same information repeatedly on those platforms, which makes it difficult to update the information later, increasing the risk of inconsistent data. Additionally, job seekers frequently maintain multiple versions of resumes tailored for different roles or industries, where only a subset of the information differs. Managing these variations manually reduces maintainability and scattered files and information. Moreover, parsing resume PDFs is inherently unreliable due to formatting variations and the lack of a standardized structure. This paper proposes a standardized protocol for maintaining a centralized repository of structured resumes that solves all the mentioned problems. The system allows users to store their professional information in a single authoritative location while enabling multiple variants of resumes tailored to specific roles or contexts using tags with minimal effort. To support interoperability across recruitment platforms and to minimize parsing issues, we introduce a standardized JSON-based schema for representing resume data, along with a sharing protocol powered by OAuth that allows recruiters and job portals to access resume information in a structured way. By eliminating the need for repeated manual data entry and unreliable document parsing, the proposed approach improves data consistency, simplifies resume management, and enables seamless integration between job seekers and recruitment platforms. It saves time by making resume sharing as easy as logging in with Google.

Keywords: Centralized Resume Management System (CRMS), Structured Resume Data, Resume Standardization, JSON Resume Schema, Resume Interoperability, Applicant Tracking Systems (ATS), Resume Parsing, Dynamic Resume Generation, Persistent Resume Links, Tag-Based Resume Versioning, Resume Sharing Protocol, Professional Profile Management, Recruitment Platforms, Structured Data Exchange, Resume Data Portability.

INTRODUCTION

The job recruitment process has undergone significant digital transformation with the emergence of online job portals, professional networking platforms, and automated applicant tracking systems [1], [2]. Despite these advancements, the management and sharing of resumes remain fragmented and inefficient. Job seekers are required to manually enter the same professional information repeatedly across multiple recruitment platforms, a process that is not only time-consuming but also makes it difficult to keep information consistent and up to date across all platforms. When a candidate's details change, there is no mechanism to propagate those changes across the platforms where their information has already been submitted, increasing the risk of inconsistent and outdated data being presented to recruiters.

Resumes are often distributed as static documents, typically in PDF or word-processing formats. Once shared with a recruiter or organization, these documents cannot reflect updates or corrections without redistributing an entirely new file. Candidates are therefore required to download, modify, and resend their resume every time their information changes, whether due to a newly acquired skill, updated work experience, or a simple correction. Beyond the inconvenience of redistribution, parsing these static files to extract structured data is inherently unreliable. Applicant tracking systems that ingest resume PDFs frequently encounter extraction errors caused by inconsistencies in formatting, layout, and design across different resume templates [3], [4]. The lack of a standardized resume structure means that the same information may be represented differently across different candidates, making automated processing inconsistent and error-prone.

Job seekers further compound this complexity by maintaining multiple versions of their resume tailored for different roles or industries. In most cases, only a subset of information differs between these versions. A candidate applying for an engineering role and a management role may share the same work history but wish to highlight different skills, positions, and achievements for each role.

Despite this overlap, managing these variations requires maintaining entirely separate documents, duplicating the majority of content and increasing the risk of inconsistencies arising between versions over time. As the number of target roles grows, so does the overhead of keeping all versions accurate and current.

To address these challenges, this paper proposes the Centralized Resume Management System (CRMS), a standardized protocol for maintaining a centralized repository of structured resume data. CRMS allows users to store their professional information in a single authoritative location and create multiple tailored resume variants from that single source using a tag-based versioning mechanism that require only minimal additional effort per variant. A standardized JSON-based schema serves as the canonical representation of resume data, eliminating the inconsistencies and parsing failures associated with unstructured document formats. The system also allows creating a traditional PDF file from the data if required. Additionally, instead of sharing static files, users share persistent tag-scoped links that always reflect the most current state of their resume data at the moment of access. To support interoperability across recruitment platforms, CRMS introduces an OAuth 2.0-based sharing protocol that allows job portals to access structured resume data with explicit user consent [5], [6]. Just as logging in with Google has made account creation seamless across the web, logging in with CRMS aims to make professional profile creation equally effortless across the recruitment ecosystem.

The remainder of this paper is organized as follows. Section 2 reviews related work on resume standardization, professional profile platforms, applicant tracking systems, and authorization frameworks. Section 3 presents the system design and architecture of CRMS, including the JSON schema, tag-based versioning mechanism, persistent link model, and OAuth sharing protocol. Section 4 describes the implementation of the system. Section 5 discusses the benefits, limitations, and future directions of the proposed approach. Section 6 concludes the paper.

RELATED WORK

Resume Representation and Standardization

Efforts to standardize resume data representation have existed for several years. The JSON Resume project introduced a community-driven, open-source initiative to create a JSON-based standard for resumes, motivated by the belief that JSON's lightweight and machine-readable nature makes it a strong fit for structured resume data [7]. While it provides a common schema covering work experience, education, and skills, it does not address multi-version management, persistent sharing, or integration with external platforms. On the enterprise side, the HR Open Standards Consortium is the only independent, non-profit, volunteer-led organization dedicated to developing a standard suite of specifications to enable human resource related data exchanges [8]. Originally founded in 1999 as the HR-XML Consortium, the organization re-emerged as HR Open Standards, reflecting its renewed mission to champion interoperability between HR technology systems. Despite these efforts, widespread adoption has been limited among individual users and smaller platforms due to complexity and lack of user-facing tooling. Our proposed schema builds on the philosophy of structured resume representation but extends it with a tagging mechanism that enables context-aware, multi-variant resume generation from a single data source.

Professional Networking and Profile Management

Platforms such as LinkedIn [9] have demonstrated the value of maintaining a centralized, persistent professional profile accessible via a stable URL. LinkedIn allows users to maintain a single profile that recruiters and employers can access at any time, always reflecting the most current information. However, LinkedIn operates as a closed ecosystem and does not provide a standardized mechanism for exporting or sharing structured profile data with third-party recruitment platforms. Furthermore, it does not support multiple tailored versions of a profile for different job roles or industries. The digital transformation of human resource management has accelerated the adoption of such platforms, yet fundamental problems in data portability and multi-platform profile management remain unresolved [1], [2]. Our system draws inspiration from the persistent profile model while addressing these limitations by introducing an open, interoperable, and multi-variant resume management approach.

Applicant Tracking Systems and Resume Parsing

Applicant Tracking Systems (ATS) are widely used by organizations to manage recruitment workflows. These systems typically ingest resumes in PDF or Word format and extract structured information through parsing algorithms [3]. However, resume parsing is inherently unreliable. The fundamental mismatch between PDF's presentation-oriented structure and the semantic data extraction needs of ATS systems creates significant challenges. Furthermore, Text extraction from PDFs can scramble layouts, merge columns incorrectly, and lose structural information [10]. The absence of a widely adopted standardized CV format worsens parsing problems, as candidates create CVs in diverse formats, layouts, and structures, each presenting unique parsing challenges. Reported parsing accuracies vary from 80% to 94% depending on the dataset and technique used, with limitations ranging from inconsistencies in datasets and formats to domain-specific biases [4]. Recent work has attempted to address these limitations through more sophisticated techniques. Bevara et al. (2025) proposed Resume2Vec, an innovative framework that utilizes transformer-based deep learning models including BERT, RoBERTa, and DistilBERT to create embeddings for resumes and job descriptions, achieving improvements of up to 15.85% in ranking quality over conventional ATS systems [11]. Similarly, Gan et al. (2024) proposed an LLM-agent-based framework for automated resume screening that demonstrated improved candidate-job matching performance over keyword-based ATS approaches [12]. While such approaches improve matching quality, they still rely on parsing unstructured documents as input. By introducing a structured JSON schema as the canonical representation of resume data, the proposed CRMS eliminates parsing entirely, making resume data directly and reliably consumable by ATS and other recruitment tools.

OAuth 2.0 and Delegated Authorization

The OAuth 2.0 authorization framework, standardized in RFC 6749 by Hardt (2012) under the Internet Engineering Task Force (IETF), enables a third-party application to obtain limited access to an HTTP service on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service [5]. It is the foundation of widely used authentication patterns such as "Login with Google" and "Login with Facebook". OpenID Connect builds on top of OAuth 2.0 as a simple identity layer, enabling clients to verify the identity of end-users based on authentication performed by an authorization server, as well as to obtain basic profile information in an interoperable and REST-like manner [6]. Apandi et al. (2023) demonstrated a practical implementation of OAuth 2.0 for Single Sign-On in an academic system, confirming the viability of the authorization code flow for multi-platform credential sharing [13].

The proposed CRMS sharing protocol is inspired by and built upon the OAuth 2.0 framework, adapting it for the specific use case of sharing structured resume data with job portals. Rather than sharing identity credentials, the system shares tagged resume data, giving users granular control over what information is disclosed to each platform.

Structured Data and AI-Driven Recruitment

The shift toward structured data in recruitment technology has been driven by the limitations of unstructured document processing. Unlike traditional methods, embedding-based approaches can handle unstructured data and adapt to various resume formats, addressing critical limitations in conventional ATS approaches [11]. However, even these advanced techniques are constrained by the quality of upstream data ingestion. One of the primary challenges in resume parsing is ensuring accuracy and consistency in data extraction across various resume formats and styles, arising from the diverse ways candidates structure their resumes, which can include varying layouts, terminologies, and levels of detail [14]. Recent work on zero-shot resume-job matching using structured prompting and semantic embeddings has demonstrated that structured data inputs significantly improve matching precision, achieving up to 87% accuracy for specific occupations [15]. Similarly, Smart-Hiring, an end-to-end NLP pipeline for CV information extraction and job matching, highlighted that when PDF layouts contain multiple columns or dense formatting, extracted text loses its original structure, negatively impacting downstream accuracy. Storing resume data in a structured JSON schema, as proposed in this paper, eliminates these upstream inconsistencies entirely, making the data immediately suitable for AI-driven matching, semantic search, and vector embedding systems without any parsing step [16].

SYSTEM DESIGN AND ARCHITECTURE

System Overview

The Centralized Resume Management System (CRMS) is designed around three core components: a backend server, a web application, and a dual-database storage layer. The backend serves as the central engine of the system, responsible for handling all resume data storage, retrieval, and OAuth-related API interactions. The web application, built using Next.js, provides the user-facing interface through which users create, manage, and share their resumes. Together, these components form a unified platform that decouples resume data from its presentation, enabling dynamic, on-demand generation of resume documents from a single structured data source.

Resume JSON Schema

At the core of CRMS is a standardized JSON schema that serves as the canonical representation of a user's professional information. The design philosophy behind the schema is to separate data from presentation. All professional details are stored in a structured, machine-readable format, while the visual rendering of the resume is handled independently at request time.

The schema is versioned via a top-level version field, allowing the protocol to evolve over time while maintaining backward compatibility. The schema is organized into the following top-level sections: `personal_details`, `work_experience`, `skills`, `projects`, `education`, `achievements`, `publications`, and `otherLists`. The `otherLists` section serves as a flexible wildcard, allowing users to define custom sections for information that does not fit into the standard categories.

A central design pattern across the schema is the embedding

of tags directly within individual data fields. Rather than applying tags at the section level alone, the schema supports tagging at multiple levels of granularity. For instance, within a `work_experience` entry, the entry itself carries tags, but individual `position`, `summary`, and `highlights` fields within that entry also carry their own independent tags. This allows users to include the same job in multiple resume versions while presenting different positions or highlights depending on the target role.

The following JSON snippet illustrates this tagging pattern within a work experience entry:

```
{
  "company": "Tech Solutions Ltd.",
  "tags": [
    { "tag": "#general" },
    { "tag": "#engineering-role" }
  ],
  "position": [
    {
      "text": "Full Stack Developer",
      "tags": [{ "tag": "#engineering-
role" }]
    },
    {
      "text": "Technical Lead",
      "tags": [{ "tag": "#management-role" }]
    }
  ],
}
```

In this example, the same company entry appears in both the `#engineering-role` and `#management-role` resume versions, but each version presents a different position title, allowing a single data entry to serve multiple tailored resume versions without duplication.

Tag-Based Resume Versioning

The tag-based versioning system is the mechanism through which CRMS enables users to maintain multiple tailored resume versions from a single data source. Every user begins with a default tag called `#general`, which represents the base version of their resume containing all their professional information.

To create a new resume version, the user creates a new tag, for example, `#marketing-role`, derived from the `#general` base. The new tag inherits all data from the base tag by default. The user then customizes this version by adding the new tag to data entries that should appear in it and removing the tag from entries that are not relevant. This approach means that users only manage the differences between versions rather than maintaining entirely separate resume documents.

At render time, the system filters all fields in the JSON document, retaining only those entries whose tags array contains the requested tag. Fields that carry no tags, such as `name`, `email`, and `date_of_birth` are considered universal and are always included regardless of the requested tag. The following snippet illustrates how the same skills section produces different outputs depending on the requested tag:

```

{
  "skills": {
    "technical": [
      {
        "name": "React.js",
        "tags": [
          { "tag": "#general" },
          { "tag": "#engineering-role" }
        ]
      },
      {
        "name": "Google Ads",
        "tags": [ { "tag": "#marketing-role" } ]
      },
      {
        "name": "SEO Optimization",
        "tags": [ { "tag": "#marketing-role" } ]
      }
    ]
  }
}

```

When the #engineering-role tag is requested, only React.js is included. When #marketing-role is requested, Google Ads and SEO Optimization are included instead. This filtering logic is applied uniformly across all sections of the schema at render time.

Persistent Dynamic Resume Links

Traditional resume sharing relies on distributing static files, typically PDFs, via email or file-sharing links. This approach suffers from a fundamental limitation. Once a file is shared, it cannot be updated. If a candidate acquires a new skill or corrects an error after sharing, the recipient continues to

[https://centralresume.me/john_doe?tag=# general](https://centralresume.me/john_doe?tag=#general) [https://centralresume.me/john_doe?tag=# marketing-role](https://centralresume.me/john_doe?tag=#marketing-role)

When this link is opened, the system retrieves the user’s JSON resume data in real time, applies the tag-based filtering for the specified tag, and dynamically generates a PDF on demand. The recipient always sees the most current version of the candidate’s information. The URL itself never changes regardless of how many times the underlying data is updated, making it suitable for inclusion in job applications, email signatures, or professional profiles.

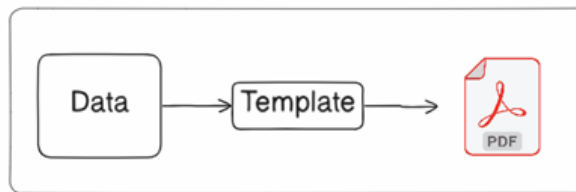


Figure-1: Resume from JSON Data.

OAuth2-Based Resume Sharing Protocol

To address the problem of repeated manual data entry across job platforms, CRMS introduces a resume sharing protocol built on the OAuth 2.0 authorization framework [5]. The protocol follows the same conceptual model as “Login with Google,” where a user authorizes a third-party platform to access a subset of their data from a trusted provider. In CRMS, this mechanism is referred to as “Login with CRMS”.

Job platforms integrate with CRMS via a standard API. Once integrated, the platform can present a “Login with CRMS” button to its users. The high-level flow proceeds as follows:

- i. Authorization Request — The user clicks “Login with CRMS” on a job platform. The platform redirects the user to the CRMS authorization server.
- ii. Tag Selection and Consent — The user is presented with their available resume tags and selects which tag they wish to share with the platform. The user explicitly consents to sharing only the data associated with the selected tag.
- iii. Token Issuance — CRMS issues an access token to the job platform scoped to the selected tag.
- iv. Data Access — The job platform uses the access token to retrieve the structured resume data filtered by the authorized tag via the CRMS API, and uses it to populate the user’s profile on the platform.

This flow ensures that users retain full control over what information is shared with each platform. A user may share their #engineering-role tag with one platform and their #management-role tag with another. Furthermore, since the job platform accesses data via a live API rather than a one time export, any updates the user makes to their CRMS profile are automatically reflected on the job platform at the next data sync, eliminating the need to manually update profiles across multiple platforms.

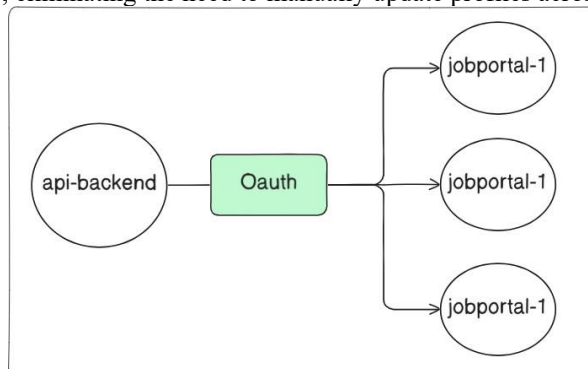


Figure-2: Resume profile sharing to job platforms.

Database Design

CRMS employs a hybrid database architecture to optimally serve the distinct storage requirements of different parts of the system. Resume data is stored in MongoDB, a document-oriented NoSQL database. This choice is well-suited to the nature of resume data. The JSON schema is hierarchical, variable in structure across users, and subject to evolution as the protocol version advances. MongoDB's flexible document model accommodates these characteristics without requiring rigid schema migrations [17], [18]. All other system data including user accounts, authentication records, OAuth tokens, tag definitions, platform integrations, and access control metadata is stored in PostgreSQL, a relational database. This data is inherently relational and benefits from the strong consistency guarantees, transactional integrity, and structured querying capabilities that PostgreSQL provides. The combination of MongoDB for document storage and PostgreSQL for relational data allows CRMS to apply the most appropriate data management model to each category of information within the system.

IMPLEMENTATION

Backend

The backend is structured around four core responsibilities. Resume data management handles CRUD operations for JSON-based resumes stored in MongoDB. Tag management enables creation, association, and derivation of tags for organizing resume data. OAuth APIs implement the CRMS authorization server, managing consent flows, token issuance, and secure data sharing with thirdparty platforms. The PDF generation service processes requests from persistent links, applies tag-based filtering, and returns dynamically generated resumes. Tag filtering is implemented at the middleware level. When a tag is specified, the system traverses the resume JSON and retains only fields associated with that tag. Untagged fields, such as personal and contact information, are treated as universal and always included. The filtered JSON is then used for PDF generation or API responses.

Web Application

The web application is built using Next.js, enabling server-side rendering of persistent resume links so recipients always receive up-to-date, fully populated pages without relying on client-side fetching. This improves reliability and compatibility across environments with limited JavaScript support.

The application includes three main interfaces. The resume editor allows users to create and manage structured resume data across all sections. The tag manager enables creation, derivation, and assignment of tags to organize data. The profile page renders a public resume view from tag-filtered JSON data, functioning as a dynamic professional profile.

PDF Generation

Dynamic PDF generation enables CRMS's persistent link model, where a stable URL always returns an up-to-date resume. It is implemented using React-PDF, which defines resume layouts as reusable React components mapped directly to the JSON schema. When a link is accessed, the server retrieves the user's resume from MongoDB, applies tag-based filtering, and passes the result to the React-PDF pipeline. Each JSON section is rendered into its corresponding component, producing a formatted PDF returned in the response.

This ensures the PDF always reflects the latest resume data without requiring manual updates.

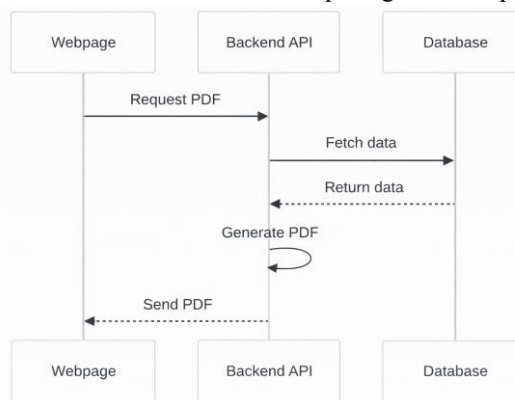


Figure-3: Sequence of dynamic pdf generation

Schema Versioning

The CRMS JSON schema includes a top-level version field to track the protocol version of each resume, enabling evolution of the standard to support new fields, use cases, and improvements to the tagging model [19].

The backend maintains backward compatibility by applying version-specific processing logic when retrieving documents, ensuring older resumes remain functional without data loss. When users edit their resumes, they are offered migration to the latest schema version, preserving existing data while incorporating new capabilities.

DISCUSSION

Benefits to Job Seekers

The proposed CRMS offers substantial practical benefits to job seekers by fundamentally simplifying how professional information is created, maintained, and shared. By storing all professional data in a single authoritative location, the system eliminates the fragmentation that arises from maintaining multiple disconnected resume files across different devices and platforms. Any update made to the central data store, such as adding a newly acquired skill, updating a job title, or correcting contact information, is immediately reflected across all resume versions and all platforms that have been granted access, without any additional effort from the user.

The tag-based versioning system further reduces the effort involved in managing multiple resume variants. Rather than maintaining entirely separate documents for different target roles, users manage a single dataset and control which entries are visible in each version through tag assignment. This approach scales naturally. A user may maintain two tags or twenty, the overhead of managing each additional version is limited to tagging the relevant data entries rather than duplicating and editing entire documents.

The persistent link model transforms resume sharing into a seamless experience. A user can include a single stable link in every job application, email signature, or professional communication, confident that the recipient will always see the most current version of their resume. This eliminates the need to re-download, reformat, and redistribute updated resume files with every application cycle.

Benefits to Recruiters and Job Platforms

Recruiters and job platforms stand to benefit significantly from the structured, standardized nature of resume data delivered by CRMS. Because all resume data is stored and transmitted as a well-defined JSON document rather than a parsed PDF, the data received by recruiters is complete, consistently structured, and free from the extraction errors that commonly affect ATS systems [3], [4], [10]. Fields such as skills, experience, and education are always present in predictable locations within the schema, enabling reliable automated processing, filtering, and ranking of candidates without the need for document parsing.

For job platforms, integration with CRMS via the OAuth-based sharing protocol substantially reduces user onboarding friction. Rather than asking new users to manually complete lengthy profile registration forms, an integrated platform can populate a user's professional profile in its entirety from a single authorized API call. This improvement in onboarding experience has direct implications for user acquisition and platform engagement. Furthermore, because the job platform accesses resume data through a live API rather than a one-time import, candidate profiles on the platform remain current as users update their information in CRMS, improving the quality and reliability of candidate data available to recruiters over time.

Benefits to AI and ML Systems

The structured JSON representation of resume data offers significant advantages for AI and machine learning systems used in recruitment. Conventional AI-driven recruitment tools, including semantic search engines, candidate ranking systems, and job matching algorithms, must first parse unstructured resume documents before any meaningful analysis can be performed [11], [15], [20]. This parsing step introduces noise, inconsistency, and data loss that propagates through the entire analysis pipeline, degrading the quality of downstream outputs.

By providing resume data in a clean, structured, and consistently formatted JSON document, CRMS eliminates the parsing step entirely. Resume fields are immediately available as structured inputs to machine learning models, vector embedding systems, and semantic search pipelines without any preprocessing [16]. The standardized schema also ensures that feature extraction is consistent across all resumes in the system, enabling more accurate model training and inference. Skills, experience levels, and other structured fields can be directly encoded as features without ambiguity, improving the precision of candidate-to-job matching algorithms and reducing bias introduced by inconsistent data representations [21].

Security and Privacy Considerations

Security and privacy are foundational concerns in any system that manages sensitive personal and professional data. CRMS addresses these concerns through the OAuth 2.0 authorization model [5], [22], which ensures that third-party job platforms never have direct access to a user's credentials or raw data store. Access is always mediated through scoped access tokens issued by the CRMS authorization server, and each token is limited in scope to the specific tag that the user authorized during the consent flow. A job platform granted access to a user's #marketing-role tag cannot access data associated with any other tag, nor can it access fields that the user has not explicitly included in that tag.

User consent is an explicit and informed step in the sharing flow. Users are presented with a clear summary of what data will be shared before authorizing any platform, consistent with the principles of data minimization and purpose limitation outlined in the General Data Protection Regulation (GDPR) [23]. Users retain the ability to revoke access granted to any platform at any time, after which the platform's access token is invalidated and further data access is prevented.

The centralized nature of CRMS also reduces the overall attack surface associated with resume data. Rather than sensitive professional information being scattered across multiple job platforms with varying security standards, users maintain a single authoritative store governed by consistent security policies. The risk of stale or inconsistent data on third-party platforms is mitigated by the live API access model, which reduces the need for platforms to retain local copies of resume data beyond what is operationally necessary.

Limitations

Despite its advantages, the proposed system has several limitations that must be acknowledged. The most significant is the adoption dependency inherent in the OAuth-based sharing protocol. The value of "Login with CRMS" is directly proportional to the number of job platforms that integrate with CRMS. In its early stages, before widespread platform adoption, users would still need to manually create profiles on non-integrated platforms, limiting the immediate practical benefit of the system. Overcoming this adoption barrier requires deliberate outreach and incentivization of job platform integration.

The centralized architecture also introduces a single point of failure. If the CRMS service experiences downtime, all persistent resume links become temporarily inaccessible and OAuth-based data sharing is interrupted. This is a meaningful concern given that resume access is often time-sensitive in the context of active recruitment. Mitigation strategies such as redundant infrastructure, content delivery networks, and cached PDF snapshots would need to be implemented at scale to address this risk.

Finally, while the standardized JSON schema is a strength of the system, it also imposes a degree of rigidity on how professional information can be represented. Highly unconventional career profiles, creative portfolios, or domain-specific professional formats may not map cleanly onto the predefined schema sections. The otherLists field partially addresses this by providing a flexible custom section, but it may not fully accommodate all edge cases in professional representation.

As the system scales to accommodate a growing user base and increasing API traffic, the current technology stack may require revision to meet performance and reliability demands. The present implementation, built on Node.js, Express, MongoDB, and PostgreSQL, is well suited for rapid development and early-stage deployment but may encounter limitations in throughput and concurrency at significant scale. Future work includes evaluating and migrating to higher-performance backend technologies where appropriate, for instance, adopting a compiled, statically typed language such as Go or Rust for performance-critical components such as the PDF generation pipeline and OAuth token processing. The hybrid database architecture is expected to remain appropriate at scale.

RESULTS

This section evaluates the Centralized Resume Management System (CRMS) across schema coverage, parsing performance, data extraction accuracy, tag filtering efficiency, and profile creation time, with comparisons to traditional ATS based approaches.

Scheme Coverage

The CRMS JSON schema directly covers 85% of standard resume fields, including all core sections, while the remaining 15% are supported through a flexible otherLists structure. This provides comparable coverage to existing standards like JSON Resume, with added extensibility and tagging capabilities.

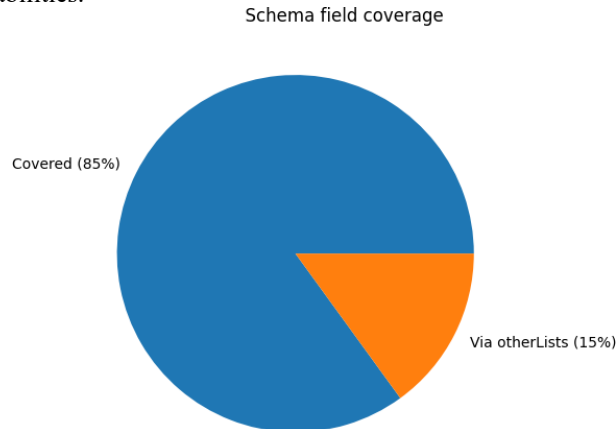


Figure-4: Resume type coverage

Parsing Performance

Traditional ATS systems require 2–8 seconds to parse resumes. CRMS eliminates parsing entirely by using structured JSON, resulting in effectively 0 ms processing time and removing associated overhead.

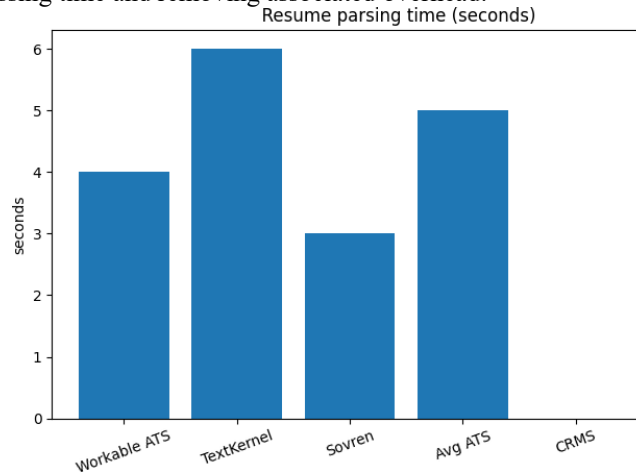


Figure-5: Parsing time comparison

Data Extraction Accuracy

Existing systems achieve 80–94% accuracy due to parsing limitations. CRMS achieves 100% accuracy by design, as data is directly authored in structured format, eliminating extraction errors.

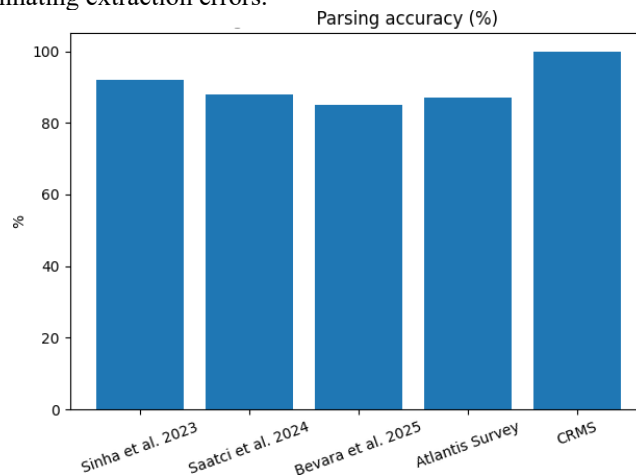


Figure-6: Parsing Accuracy

Profile Creation Time

Manual profile creation takes 5 minutes, whereas CRMS completes the process in 5 seconds via OAuth, reducing time by approximately 98%.

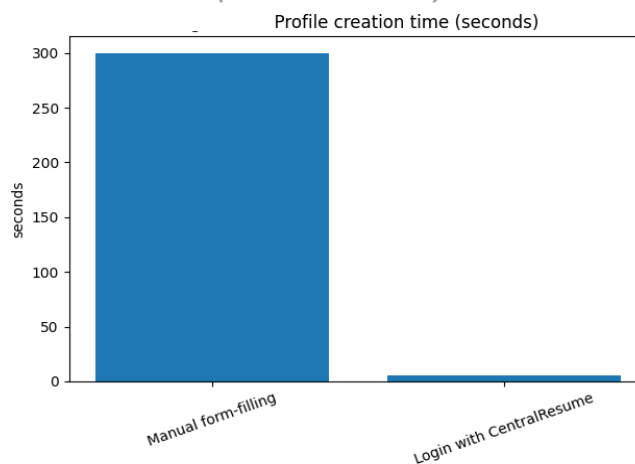


Figure-7: Profile creation time.

CONCLUSION

Creating and maintaining job profiles across multiple recruitment platforms is repetitive, error-prone, and difficult to manage consistently. The need to maintain multiple tailored resumes and redistribute updated PDF versions further increases complexity, while the lack of standardized structure in such documents leads to unreliable data extraction by applicant tracking systems.

This paper introduced the Centralized Resume Management System (CRMS), a standardized approach for managing structured resume data through a unified repository. By using a JSON-based schema and a tag-driven versioning mechanism, CRMS enables efficient creation of multiple resume variants with minimal overhead. Persistent, tag-scoped links ensure access to the most up-to-date information, while an OAuth

2.0-based sharing protocol allows secure, consent-driven integration with recruitment platforms.

Overall, CRMS reduces redundancy, improves data consistency, and streamlines interoperability across the recruitment ecosystem, enabling a more seamless and efficient approach to professional profile management.

REFERENCES

- [1] V. Baranyi, "Systematic literature review on the digital transformation of the personnel selection process," *Journal of Industrial and Organizational Psychology*, 2025.
- [2] T. Huang, "Exploring human resource management digital transformation in the digital age," *Journal of Knowledge Economy*, vol. 14, pp. 1–17, 2023.
- [3] A. K. Sinha, M. A. K. Akhtar, and M. Kumar, "Automated resume parsing and job domain prediction using machine learning," *Indian Journal of Science and Technology*, vol. 16, no. 26, pp. 1967–1974, 2023.
- [4] M. Saatçı, R. Kaya, and R. Ünlü, "Resume screening with natural language processing (NLP)," *Alphanumeric Journal*, vol. 12, no. 2, pp. 121–140, 2024.
- [5] D. Hardt, "The OAuth 2.0 Authorization Framework," RFC 6749, 2012.
- [6] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, and C. Mortimore, "OpenID Connect Core 1.0 Incorporating Errata Set 1." 2014.
- [7] JSON Resume, "JSON Resume — The Open Source Initiative to Create a JSON-Based Standard for Resumes." 2013.
- [8] HR Open Standards Consortium, "HR Open Standards — The Global Community for HR Data Interoperability."
- [9] LinkedIn Corporation, "LinkedIn — Professional Networking Platform."
- [10] Anonymous, "The Resume Parsing Crisis." 2025.
- [11] R. V. K. Bevara et al., "Resume2Vec: Transforming Applicant Tracking Systems with Intelligent Resume Embeddings for Precise Candidate Matching," *Electronics*, vol. 14, no. 4, p. 794, 2025.
- [12] C. Gan, Q. Zhang, and T. Mori, "Application of LLM Agents in Recruitment: A Novel Framework for Automated Resume Screening," *Journal of Information Processing*, vol. 32, pp. 881–893, 2024.
- [13] T. H. Apandi, M. Iqbal, R. Piarna, D. Vernanda, and A. R. M. H. N. Asegaff, "Web-Based Single Sign-On Approach for the Authorization Server Using OAuth 2.0 Protocol in Academic System," *Appissode: Application, Information System and Software Development Journal*, vol. 1, no. 1, pp. 20–26, 2023.
- [14] Anonymous, "Advancements in Automated Resume Assessment: A Comprehensive Survey." 2024.
- [15] A. Poirier, "Zero-Shot Resume–Job Matching with LLMs via Structured Prompting and Semantic Embeddings," *Electronics*, vol. 14, no. 24, p. 4960, 2025.
- [16] V. Mittal, P. Mehta, D. Relan, and G. Gabrani, "Methodology for Resume Parsing and Job Domain Prediction," *Journal of Statistics and Management Systems*, vol. 23, no. 7, pp. 1265–1274, 2020.
- [17] S. Hamouda and Z. Zainol, "A Flexible Schema for Document Oriented Database (SDOD)," in *Proc. 2017 International Conference on Big Data Innovations and Applications (Innovate-Data)*, 2017, pp. 1–7.
- [18] A. Hillenbrand and others, "NoSQL Schema Evolution and Data Migration: State-of-the-Art and Opportunities," in *Proc. EDBT*, 2020, pp. 1–10.
- [19] A. Wright, H. Andrews, B. Hutton, and G. Dennis, "JSON Schema: A Media Type for Describing JSON Documents." 2020.
- [20] Anonymous, "Smart-Hiring: An Explainable End-to-End Pipeline for CV Information Extraction and Job Matching." 2025.
- [21] M. Fernández and A. Gómez, "Job Vacancy Ranking with Sentence Embeddings, Keywords, and Named Entities," *Information*, vol. 14, no. 8, p. 468, 2023.
- [22] S. S. M. Rahman and others, "OAuth 2.0: A Framework to Secure the OAuth-Based Service for Packaged Web Application," *Innovative Perspectives on Interactive Communication Systems and Technologies*. IGI Global, pp. 92–139, 2020.
- [23] European Parliament and Council of the European Union, "Regulation (EU) 2016/679 — General Data Protection Regulation." 2016.