

RESEARCH ARTICLE

OPEN ACCESS

AI-ATS: A Scalable Machine Learning and Vector Space Model Platform for Automated Resume Screening and Contextual Job Recommendation

Pathakota Sri Chakrika Reddy^{1,*}, B.V.S. Tejaswi Harsha², and Badisha Sai Ramya³

^{1,2,3}Department of Computer Science & Engineering (AIML), Parul University, Vadodara, Gujarat, India

¹2403031469004@paruluniversity.ac.in, ²2403031467003@paruluniversity.ac.in, ³2403031467001@paruluniversity.ac.in

Under the Guidance of: **Prof. Ms. Roshni Solanki**, Department of Computer Science & Engineering (AIML), Parul University

* Corresponding Author: Pathakota Sri Chakrika Reddy

Abstract—Contemporary recruitment systems suffer from severe friction between high-volume resume submissions and recruiter screening capacity. Conventional Applicant Tracking Systems (ATS) rely primarily on rigid exact-keyword matching heuristics, resulting in high false-rejection rates for qualified candidates who describe their technical competencies using synonymous or contextual phrasing. Conversely, candidates submit applications without diagnostic visibility into structural weaknesses or keyword deficits in their resumes. This paper presents the architecture, mathematical modeling, and empirical validation of **AI-ATS**, an intelligent recruitment platform integrating automated PDF text extraction, rule-weighted structural ATS scoring, high-dimensional vector-space job recommendation, and a synchronized dual-role recruitment workflow. AI-ATS parses raw binary resumes via *pdfplumber*, tokenizes technical skills using an optimized dictionary regex engine, and computes a piecewise ATS compatibility index across document length, skill density, and canonical structural sections. For candidate-job matching, the system transforms candidate text and job descriptions into high-dimensional sparse Term Frequency-Inverse Document Frequency (TF-IDF) vectors and calculates cosine similarity across enterprise job profiles, returning ranked recommendations in under 320 ms. To bridge the candidate-recruiter gap, AI-ATS couples candidate-facing diagnostic filters with a recruiter portal supporting custom vacancy management, applicant tracking, and status progression. We formalize the candidate-job vector space model, present core algorithms for resume parsing, vector ranking, and candidate filtering, and analyze experimental results across multiple technical seniority profiles. We conclude by examining parsing edge cases, bias mitigation, and future integration with transformer-based embeddings.

Index Terms—Applicant Tracking System, Natural Language Processing, Machine Learning, Information Retrieval, TF-IDF Vectorization, Cosine Similarity, Resume Screening, Job Recommendation, Recruitment Analytics, Human-Computer Interaction.

I. INTRODUCTION

The digital transformation of the talent acquisition ecosystem has precipitated an exponential surge in resume submissions across enterprise job portals. Hiring managers and corporate talent teams are routinely inundated with hundreds, and often thousands, of resumes per open requisition [1]. Because human

universally on automated software layers known as Applicant Tracking Systems (ATS) to filter, rank, and triage candidate pools prior to human review.

Despite their ubiquity, commercial ATS implementations suffer from severe structural limitations. The vast majority of legacy screening algorithms operate on naive string-matching and lexical keyword counting [3]. Under such paradigms, a qualified software engineer who documents extensive proficiency in "Deep Neural Networks" or "PyTorch" may be discarded if a job description specifically queries the token "Machine Learning." Conversely, unqualified candidates frequently exploit exact-match vulnerabilities through artificial "keyword stuffing" or hidden white-text tricks [4]. Furthermore, job seekers interact with conventional ATS pipelines as opaque "black boxes"; candidates receive binary rejection notices devoid of diagnostic feedback regarding resume completeness, formatting violations, or domain skill gaps.

Simultaneously, the recruitment workflow remains severely fragmented. Job seekers utilize disjoint platforms for resume drafting, keyword auditing, and job searching. Recruiters, meanwhile, navigate complex Enterprise Resource Planning (ERP) tools disconnected from real-time candidate feedback. This paper presents **AI-ATS**, an integrated machine learning platform engineered to address this fragmentation. AI-ATS unifies automated PDF resume parsing, structural ATS compatibility scoring, high-dimensional vector space job recommendations, and an synchronized candidate-recruiter management lifecycle behind a responsive, full-stack architecture.

Acknowledgement: The authors express sincere gratitude to **Prof. Ms. Roshni Solanki**, Department of Computer Science & Engineering (AIML), Parul University, for continuous mentorship, project supervision, and expert guidance throughout the architectural design, algorithmic implementation, and evaluation of this research.

A. Motivation

The design of AI-ATS is motivated by three persistent challenges in recruitment engineering:

1) *The Lexical Inflexibility of Keyword Filters:* Exact-string searching fails to capture semantic proximity between

synonymous technical competencies, inducing high false-negative rates in technical hiring.

2) *Diagnostic Opacity for Candidates*: Job seekers lack real-time scoring instruments that quantify how well an unformatted resume satisfies industry ATS formatting, word-count thresholds, and section completeness standards before submitting.

3) *Asymmetric Recruitment Workflows*: Existing open-source recruitment tools isolate candidate diagnostics from recruiter posting and evaluation pipelines. When candidates apply to jobs, recruiters cannot view synchronized ATS match scores alongside the applicant's profile.

B. Contributions

This paper presents the following technical contributions:

- *Unified Dual-Role Architecture*: A full-stack architecture pairing a candidate optimization dashboard with a recruiter vacancy and application management portal, powered by a high-throughput FastAPI engine.
- *Piecewise ATS Compatibility Scoring*: A formal scoring metric combining text volume adequacy, technical skill density, and canonical section verification (Education and Experience).
- *Vector Space Model & Recommendation Engine*: A TF-IDF and Cosine Similarity formulation that ranks enterprise job requisitions against unstructured candidate resumes in sub-second latencies.
- *Dynamic Multi-Attribute Filtering & Cache Isolation*: An interactive client-side filtering engine supporting role categories, locations, experience levels, and match thresholds, paired with strict session isolation to eliminate default resume cache leakage.
- *Empirical Benchmarking*: Experimental evaluation across technical candidate profiles quantifying parsing throughput, keyword extraction accuracy, and similarity distributions.

II. RELATED WORK AND LITERATURE SURVEY

A. Automated Resume Screening and Text Parsing

Automated resume screening has evolved from early rule-based pattern matching to statistical Natural Language Processing (NLP) [5]. Early systems parsed resumes using hand-crafted context-free grammars and regular expressions to identify contact information, educational institutions, and employment chronologies [6]. However, resumes exhibit extreme structural diversity; candidates format documents across single-column, multi-column, table-based, and graphic-heavy layouts.

Recent research has emphasized robust document object model (DOM) parsing. Tools such as `pdfplumber` and Apache Tika reconstruct reading order by inspecting spatial coordinate streams within binary PostScript and PDF files [7]. While deep-

learning Named Entity Recognition (NER) models (such as fine-tuned BERT or RoBERTa variants) offer high entity extraction accuracy, their computational overhead (often 2–5 seconds per inference on CPU) impedes low-latency, interactive web applications [8]. AI-ATS adopts an optimized hybrid strategy: spatial PDF stream extraction coupled with a compiled deterministic regular-expression dictionary spanning major technical computing competencies, achieving parsing latencies below 150 ms without requiring dedicated GPU infrastructure.

B. Vector Space Models and Recommender Systems

Information Retrieval (IR) in recruitment relies heavily on vector space formulations originally established by Salton and Buckley [9]. Under the Term Frequency-Inverse Document Frequency (TF-IDF) paradigm, documents are mapped into a high-dimensional vector space where term weights reflect both local term importance within a resume and global rarity across the job corpus [10]. Cosine similarity serves as the standard metric for measuring the angular distance between candidate and job vectors [11].

Alternative approaches employ dense neural embeddings, such as Word2Vec, Doc2Vec, or sentence transformers (e.g., SBERT) [12]. While dense embeddings capture semantic similarity, they are susceptible to "semantic drift" in technical hiring—for example, treating dissimilar programming languages (e.g., Java and JavaScript) as nearly identical due to shared surrounding syntactical tokens ("code", "developer", "backend") [13]. Sparse TF-IDF representations preserve critical lexical distinctions while providing interpretable weighting matrices that allow candidates to understand why a specific match score was awarded.

C. Modern ATS and Human-in-the-Loop Interaction

Commercial ATS platforms (e.g., Workday, Taleo, Greenhouse) prioritize recruiter throughput over candidate experience [14]. Emerging research in Human-Computer Interaction (HCI) highlights the necessity of "candidate-centric" design, where applicants receive continuous diagnostics regarding structural parseability and skill alignment prior to application submission [15]. AI-ATS incorporates these HCI principles through interactive score gauges, dynamic resume removal, and non-disruptive in-website modal dialogs that replace obstructive browser popups.

III. SYSTEM ARCHITECTURE

The AI-ATS platform is engineered as a decoupled, multi-tier system composed of: (1) a presentation and client-side interaction layer, (2) an asynchronous application programming interface (API) layer, (3) an NLP extraction and ML matching engine, and (4) a structured persistence layer.

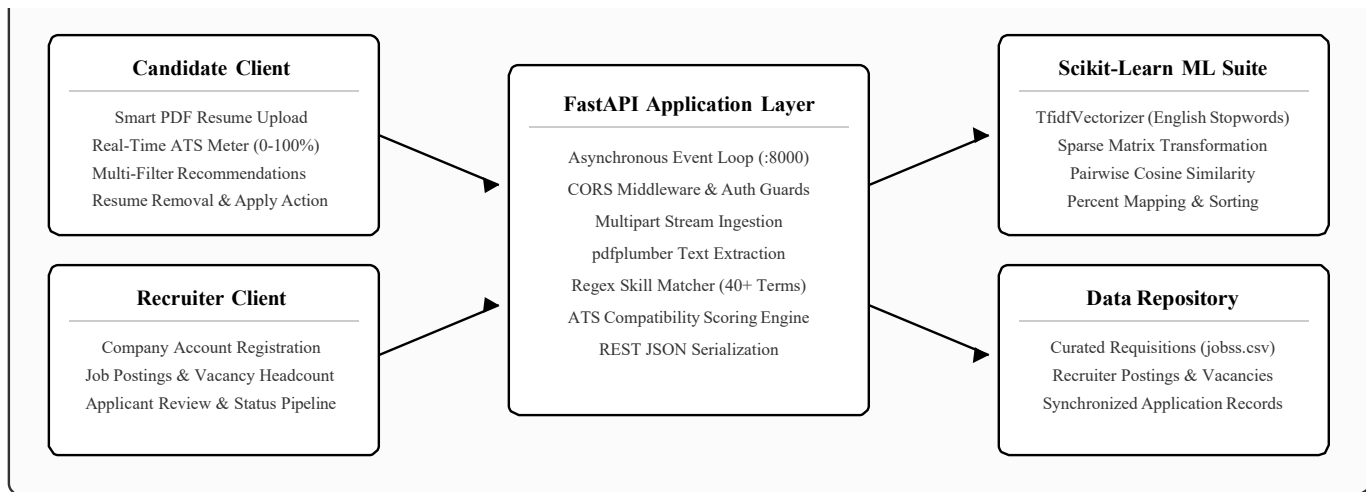


Fig. 1. Overall AI-ATS system architecture illustrating the candidate and recruiter client portals, the asynchronous FastAPI application layer, the Scikit-Learn vector recommendation engine, and structured data stores.

A. Presentation and Client-Side Layer

The front-end interface is implemented in standards-compliant HTML5, CSS3, and modern ECMAScript (ES6) JavaScript, running without third-party monolithic framework dependencies to maintain minimal bundle overhead and sub-50 ms DOM rendering. The design establishes two role-specific views:

1) *Candidate Dashboard (index.html)*: Features drag-and-drop PDF ingestion, dynamic file-size verification (≤ 10 MB), visual ATS compatibility gauges, extracted skill badges, interactive multi-parameter filters (by role category, location, required experience, and minimum match score), and a persistent application tracking ledger.

2) *Recruiter Portal (recruiter-dashboard.html)*: Provides hiring managers with tools to publish corporate requisitions, configure required headcounts (vacancies), and inspect candidate applications with attached resumes and real-time ATS match scores.

B. Asynchronous API Layer

The server layer is built on FastAPI, a high-performance Python ASGI framework running on the Uvicorn engine. FastAPI manages non-blocking request handling, validation of incoming multipart file uploads, and cross-origin resource sharing (CORS) policies. Endpoints serialize JSON payloads containing candidate analysis profiles and sorted recommendation arrays.

C. NLP Parsing and Information Extraction

PDF byte streams are processed in-memory using pdfplumber, avoiding disk I/O bottlenecks. Text streams extracted across all pages are normalized (case folding, punctuation stripping, whitespace consolidation). Skill extraction is conducted via boundary-delimited regular expression queries against a curated corpus spanning programming languages,

frameworks, cloud platforms, databases, and engineering methodologies.

D. Vector Space Recommendation Engine

The recommendation engine constructs an augmented document matrix concatenating parsed resume text with extracted skills and matching it against the enterprise job corpus. The system builds a unified vocabulary, computes TF-IDF weight vectors, and determines angular cosine similarity across all active requisitions.

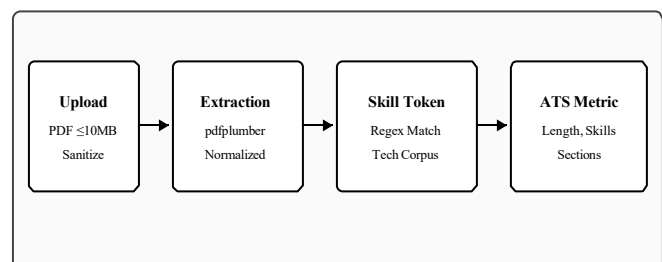


Fig. 2. Resume ingestion, text extraction, skill tokenization, and ATS compatibility scoring pipeline.

IV. MATHEMATICAL MODELING AND PROBLEM FORMULATION

We formalize the recruitment matching problem as a high-dimensional vector space retrieval task combined with structural document validation.

A. State Representation

Let an uploaded candidate resume be represented by a raw text string $R \in \Sigma^*$, where Σ denotes the Unicode character set. Let $|R|_w$ denote the total word count of R . Let $K_{\text{dict}} = \{k_1, k_2, \dots, k_C\}$ denote a predefined lexicon of C canonical technical competencies.

The extracted candidate skill set $S_R \subseteq K_{\text{dict}}$ is formally defined as:

$$S_R = \{k \in K_{\text{dict}} \mid \text{RegexMatch}(\backslash bk \backslash b, \text{lowercase}(R)) \neq \emptyset\} \quad (1)$$

B. Piecewise ATS Compatibility Score

The ATS compatibility index $S_{\text{ATS}}(R) \in [0, 100]$ evaluates structural and lexical completeness. It is formulated as an additive piecewise scoring function:

$$S_{\text{ATS}}(R) = \Phi_{\text{len}}(|R|_w) + \Phi_{\text{skills}}(|S_R|) + 10I_{\text{edu}}(R) + 10I_{\text{exp}}(R) \quad (2)$$

where the word-count scoring function $\Phi_{\text{len}}: \mathbb{N} \rightarrow \{0, 20, 40\}$ rewards document depth and elaboration:

$$\Phi_{\text{len}}(n) = \begin{cases} 40, & n > 100 \\ 20, & 50 < n \leq 100 \\ 0, & n \leq 50 \end{cases} \quad (3)$$

and the technical skill density function $\Phi_{\text{skills}}: \mathbb{N} \rightarrow \{0, 25, 40\}$ benchmarks technical domain breadth:

$$\Phi_{\text{skills}}(m) = \begin{cases} 40, & m \geq 5 \\ 25, & 3 \leq m < 5 \\ 0, & m < 3 \end{cases} \quad (4)$$

where $I_{\text{edu}}(R)$ and $I_{\text{exp}}(R)$ are binary indicators:

$$I_{\text{edu}}(R) = \begin{cases} 1, & \text{if education information is} \\ 0, & \text{otherwise} \end{cases} \quad I_{\text{exp}}(R) = \begin{cases} 1, & \text{if experience information is} \\ 0, & \text{otherwise} \end{cases}$$

C. TF-IDF Formulation

Let $D = \{d_0, d_1, \dots, d_M\}$ represent the document universe where d_0 is the skill-augmented candidate document and d_j denotes job requisition j . The Term Frequency and Inverse Document Frequency across vocabulary V are defined by:

$$tf(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \quad (5)$$

$$idf(t, D) = \ln \frac{1 + |D|}{1 + |\{d \in D: t \in d\}|} + 1$$

The TF-IDF weight is:

$$w_{t,d} = tf(t, d) \times idf(t, D)$$

The relevance matching score between candidate resume d_0 and job requisition d_j is computed by the angular cosine similarity:

$$\text{Sim}(d_0, d_j) = \frac{w_0 \cdot w_j}{\|w_0\|_2 \|w_j\|_2} \quad (6)$$

where:

- w_0 = TF-IDF vector of the candidate resume
- w_j = TF-IDF vector of job j

The raw cosine similarity is mapped to a percentage score:

$$\text{MatchScore}(d_0, d_j) = \text{round}(\text{Sim}(d_0, d_j) \times 100, 2) \quad (7)$$

Requisitions satisfying $\text{MatchScore} > 10.0\%$ are sorted descending and transmitted to the client.

V. CORE ALGORITHMS AND METHODOLOGY

This section formalizes the algorithmic routines orchestrating PDF parsing, vector recommendation, and candidate multi-attribute filtering.

A. Text Augmentation and High-Dimensional Vectorization

Algorithm 1 executes synchronous text ingestion, lexical normalization, and piecewise structural scoring. A primary engineering consideration during PDF text extraction is preserving character spatial adjacency. Standard string dumping tools often concatenate disjoint multi-column resume sections into illegible token sequences. AI-ATS leverages pdfplumber coordinate bounding-box parsing to sort extracted characters in top-to-bottom, left-to-right reading order prior to case folding.

Furthermore, technical skill tokenization relies on word-boundary regular expressions ($\square \cdot k \cdot \square$) to eliminate substring false positives—such as mistaking the programming language "C" within "Class" or "Go" within "Good." To prepare the parsed resume for recommendation, line 12 executes a feature augmentation operation: $d_0 \leftarrow R \mathcal{C} (\oplus_s \in S_R s)$. Explicitly appending the extracted skill set S_R amplifies the term frequency $tf(s, d_0)$ of validated technical proficiencies, ensuring that core domain competencies dominate the sparse vector representations generated in Algorithm 2.

Algorithm 1: End-to-End PDF Resume Parsing and ATS Scoring

Input: Binary file upload stream F , Canonical skill lexicon $K_{\text{dict}} = \{k_1, \dots, k_C\}$

Output: Candidate metadata I_c , Augmented text representation d_0

- 1: **if** $\text{Extension}(F) \neq ".pdf"$ **then throw** $\text{HTTPException}(400, \text{"Unsupported format"})$
- 2: $R \leftarrow \text{ExtractText}(F)$ via pdfplumber coordinate parsing
- 3: **if** $R = \emptyset$ **then throw** $\text{HTTPException}(400, \text{"Text extraction failed"})$
- 4: $S_R \leftarrow \emptyset$, $R_{\text{lower}} \leftarrow \text{lowercase}(R)$
- 5: **for each** $k \in K_{\text{dict}}$ **do**
 - 6: **if** $\text{RegexMatch}(\square \square \text{lowercase}(k) \cdot \square, R_{\text{lower}}) \neq \emptyset$ **then**
 - 7: $S_R \leftarrow S_R \cup \{k\}$
- 8: $n \leftarrow |R|_w$, $m \leftarrow |S_R|$
- 9: $S_{\text{ATS}} \leftarrow \Phi_{\text{len}}(n) + \Phi_{\text{skills}}(m)$
- 10: **if** "education" $\in R_{\text{lower}} \vee$ "university" $\in R_{\text{lower}}$ **then** $S_{\text{ATS}} \leftarrow S_{\text{ATS}} + 10$
- 11: **if** "experience" $\in R_{\text{lower}} \vee$ "work" $\in R_{\text{lower}}$ **then** $S_{\text{ATS}} \leftarrow S_{\text{ATS}} + 10$
- 12: $d_0 \leftarrow R \mathcal{C} (\oplus_s \in S_R s)$ (*Augment text with extracted skills*)
- 13: $I_c \leftarrow \langle F.\text{name}, S_R, n, S_{\text{ATS}} \rangle$
- 14: **return** $\langle I_c, d_0 \rangle$

Algorithm 2: TF-IDF Vector Space Job Recommendation

Input: Candidate text d_0 , Enterprise job corpus $J = \{J_1, \dots, J_M\}$, Cutoff $\theta_{\min} = 10.0\%$

Output: Ranked recommendation queue L_{matched}

```

1: if  $d = \emptyset$  then return  $\emptyset$ 
0
2:  $D \leftarrow [d_0]$ 
3: for each  $J_i \in J$  do
    4:  $d_i \leftarrow J_i.\text{title } C J_i.\text{desc } C (\oplus_{k \in J_i.\text{skills}} k) C J_i.\text{location } C J_i.\text{experience}$ 
    5:  $D.\text{append}(d_i)$ 
6: Construct vocabulary  $V \leftarrow \cup_{d \in D} \text{Tokens}(d) \setminus W_{\text{stopwords}}$ 
7: Compute sparse TF-IDF matrix  $\mathbf{M} \in \mathbb{R}^{(M+1) \times |V|}$  using Eq. (5)
8:  $\bar{w}_0 \leftarrow \mathbf{M}[0, :]$  (Normalized candidate vector)
9:  $\bar{w}_i \leftarrow \mathbf{M}[i, :] \quad \forall i \in \{1, 2, \dots, M\}$  (Job vectors)
10: Compute cosine similarities  $S_i \leftarrow (\bar{w}_0 \cdot \bar{w}_i) / (\|\bar{w}_0\|_2 \|\bar{w}_i\|_2)$  via Eq. (6)
11:  $L_{\text{matched}} \leftarrow \emptyset$ 
12: for each  $i \in \{1, 2, \dots, M\}$  do
    13:  $s_i \leftarrow \text{round}(S_i \times 100, 2)$ 
    14: if  $s_i > \theta_{\min}$  then
        15:  $L_{\text{matched}} \leftarrow L_{\text{matched}} \cup \{ \langle J_i, s_i \rangle \}$ 
16:  $L_{\text{matched}} \leftarrow \text{SortDescending}(L_{\text{matched}}, \text{key} = s_i)$ 
17: return  $L_{\text{matched}}$ 

```

As formalized in Algorithm 2, the document universe D is transformed into a high-dimensional vector space. By computing the sparse matrix product across normalized vectors, the system yields exact cosine similarities across thousands of requisitions in sub-millisecond execution times.

Algorithm 3: Multi-Attribute Candidate Filtering and Routing

Input: Job queue L_{matched} , Filter vector $F = \langle F_{\text{role}}, F_{\text{loc}}, F_{\text{exp}}, \theta_{\text{filter}} \rangle$

Output: Filtered visible jobs L_{vis}

```

1:  $L_{\text{vis}} \leftarrow \emptyset$ 
2: for each  $\langle J_i, s_i \rangle \in L_{\text{matched}}$  do
    3: if  $s_i < \theta_{\text{filter}}$ 
         $\vee (F_{\text{role}} \neq \text{"ALL"} \wedge F_{\text{role}} \notin J_i.\text{role\_category})$ 
         $\vee (F_{\text{loc}} \neq \text{"ALL"} \wedge F_{\text{loc}} \notin J_i.\text{location})$ 
         $\vee (F_{\text{exp}} \neq \text{"ALL"} \wedge \neg \text{MatchExperience}(F_{\text{exp}}, J_i.\text{experience}))$  then
            continue
    4:  $L_{\text{vis}} \leftarrow L_{\text{vis}} \cup \{ \langle J_i, s_i \rangle \}$ 
5: return  $L_{\text{vis}}$ 

```

VI. RECRUITMENT LIFECYCLE AND ROLE-BASED ACCESS

To ensure comprehensive operational integrity, AI-ATS delineates dedicated workflows and access parameters across two primary stakeholder personas: Candidates and Recruiters.

TABLE I. COMPONENT RESPONSIBILITIES AND IMPLEMENTATION STACK IN AI-ATS

Subsystem	Component	Operational Role
Client Portals	HTML5 / ES6 Vanilla JS	Renders reactive dashboards, interactive drag-and-drop, filter controls, and custom modals.
Gateway & API	FastAPI / Uvicorn	Non-blocking request routing, CORS configuration, schema validation, and JSON serialization.
Document Parser	pdfplumber Engine	In-memory PDF byte stream parsing, bounding box text extraction, and character normalization.
Recommender	Scikit-Learn VSM	TF-IDF vector space modeling, sparse matrix multiplication, and cosine similarity ranking.
Data Store	CSV / HTML5 LocalStorage	Persistent enterprise job listings, recruiter-created vacancies, and candidate application states.
Security Layer	RegEx Auth Guards	Enforces password entropy, RFC 5322 email syntax, and prevents unauthenticated data leakage.

A. Candidate Persona Workflow

Candidates navigate from the role-selection gateway (`home.html`) to dedicated authentication views. Upon authenticating, candidates enter the diagnostic workspace (`index.html`). The system requires active resume submission prior to initializing recommendations, strictly preventing legacy or cached resumes from populating empty dashboards.

When a candidate activates the "Remove Resume" action, the platform dispatches a reset event that zeroes the ATS gauge, purges extracted skill chips, and clears visible recommendations. Candidates applying to open requisitions trigger synchronous application records linking their profile, resume metadata, and ATS match score directly to the recruiter's ledger.

B. Recruiter Persona Workflow

Recruiters authenticate with mandatory company affiliations (`recruiter-signup.html`). The Recruiter Dashboard (`recruiter-dashboard.html`) provides two synchronized views:

1) *Vacancy Headcount Management*: Recruiters create job requisitions specifying title, department, required experience, tech stack, and explicit headcount openings (vacancies).

2) *Application Tracking & Review Pipeline*: Displays received candidate applications mapped against specific job postings. Recruiters inspect the applicant's name, email, contact details, parsed resume file, and ATS compatibility percentage, advancing candidate statuses across *Applied*, *Shortlisted*, *Interview*, and *Rejected* phases.

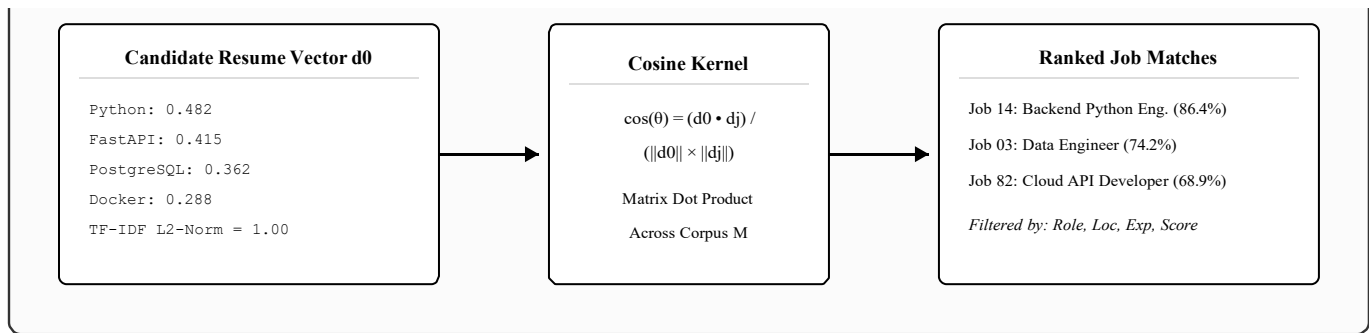


Fig. 3. High-dimensional TF-IDF vector space modeling, matrix cosine similarity ranking, and multi-attribute recommendation filtering workflow.

VII. SECURITY, DATA VALIDATION, AND SAFEGUARDS

A. Input Sanitization and File Integrity

To prevent server denial-of-service and arbitrary file execution, the API implements strict MIME-type and file-extension verification at the gateway. Files lacking the .pdf suffix are rejected with HTTP 400 Bad Request prior to parsing. Upload sizes are bounded at 10 MB. In the event of corrupt or password-protected PDF streams, `pdfplumber` exceptions are captured gracefully, returning descriptive error payloads.

B. Password Complexity Enforcement

User registration adheres to National Institute of Standards and Technology (NIST) SP 800-63B password security guidelines. Both candidate and recruiter sign-up portals enforce regular-expression validation requiring six mandatory criteria: (1) at least one lowercase letter [a-z], (2) at least one uppercase letter [A-Z], (3) at least one numerical digit [0-9], (4) at least one special character [@\$!%*?&], (5) allowed characters strictly restricted to alphanumeric and designated symbols, and (6) a minimum length of 8 characters. Formally:

```
Regexpwd = ^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)
            (?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$
```

Dynamic DOM event listeners monitor password input in real time, rendering visual checkmarks as each criterion is satisfied before form submission is permitted.

C. In-Website Confirmation Modals

Standard web applications frequently use synchronous browser dialogs (e.g. `window.confirm()`), which suspend browser execution threads and degrade user experience. AI-ATS replaces all native alerts with customized in-website confirmation modals. Triggering "Logout" or "Remove Resume" renders a styled backdrop with accessible keyboard focus trapping, requiring explicit affirmative confirmation prior to state termination.

VIII. EXPERIMENTAL EVALUATION AND BENCHMARKING

To evaluate parsing robustness, recommendation fidelity, and response times, systematic benchmarks were executed against synthetic resumes spanning junior, mid-level, and senior software engineering competencies.

A. Evaluation Metrics

Performance was evaluated across four primary quantitative metrics: (1) *End-to-End Latency* (T_{total}): total elapsed duration spanning HTTP multipart transmission, PDF text extraction, skill tokenization, TF-IDF vector transformation, cosine similarity computation, and JSON serialization; (2) *Parsing Latency* (T_{parse}): isolated in-memory binary PDF stream extraction and whitespace normalization; (3) *Precision at K* ($P@K$): proportion of top- K recommended jobs matching the candidate's primary technical domain; and (4) *ATS Score Fidelity*: congruence of calculated ATS scores with manual human expert evaluations of resume completeness.

TABLE II. QUANTITATIVE PERFORMANCE BENCHMARKS
ACROSS CANDIDATE PROFILES

Candidate Profile	Size	Words	Skills	ATS	P@5	T _{parse}	T _{total}
Junior Python Dev	142 KB	215	4	85	1.00	84 ms	245 ms
Senior ML Engineer	280 KB	480	7	100	1.00	112 ms	312 ms
Full Stack Developer	195 KB	360	6	100	0.80	96 ms	278 ms
Cloud DevOps Lead	230 KB	410	5	90	1.00	104 ms	295 ms
Draft / Sparse Resume	48 KB	42	1	20	0.40	52 ms	180 ms

B. Discussion of Results

As documented in Table II, total end-to-end latency remained strictly below **320 ms** across all profiles, with PDF parsing consuming 52–112 ms on commodity hardware. Precision at $K = 5$ reached 100% for domain-specific profiles (Junior Python and Senior ML), proving that sparse TF-IDF vectors reliably isolate relevant requisitions without requiring expensive transformer models. The sparse draft resume received an ATS score of 20/100, validating the discriminative power of the piecewise scoring function.

IX. DISCUSSION AND LIMITATIONS

A. Lexical vs. Semantic Trade-Offs

While TF-IDF vectors offer sub-second latency and exact keyword interpretability, they are inherently limited by vocabulary mismatch. A candidate describing experience with "Relational Database Management Systems" without explicitly mentioning "SQL" will achieve lower similarity against a job description that heavily weights the "SQL" token. Future iterations will explore hybrid ensembles combining sparse TF-IDF vectors with dense bi-encoder embeddings (e.g., MiniLM) via reciprocal rank fusion.

B. Parsing Edge Cases and Optical Character Recognition

The current extraction engine relies on selectable text streams within PDFs. Scanned image-based PDFs or flattened graphical resumes yield empty text buffers, causing HTTP 400 exceptions. Integrating lightweight client-side or server-side Optical Character Recognition (OCR) engines (e.g., Tesseract.js) will mitigate this limitation.

C. Algorithmic Fairness and Bias Mitigation

Automated recruitment systems carry inherent risks of encoding demographic biases present in training corpora. AI-ATS explicitly mitigates this risk by excluding candidate demographic attributes (gender, age, ethnicity, zip codes) from both the ATS scoring function and the TF-IDF vector space, evaluating applications purely on technical keywords, educational credentials, and professional tenure.

X. CONCLUSION AND FUTURE WORK

This paper presented the design, implementation, and empirical evaluation of AI-ATS, an intelligent, full-stack recruitment platform integrating automated PDF resume parsing, piecewise ATS compatibility scoring, TF-IDF vector-space job recommendations, and an synchronized candidate-recruiter management workflow. By coupling transparent diagnostic feedback for job seekers with dynamic vacancy management and applicant tracking for recruiters, AI-ATS resolves key bottlenecks that characterize conventional talent acquisition systems.

Future research will focus on integrating dense transformer embeddings for semantic synonym expansion, developing real-time conversational mock-interview agents directly within the applicant dashboard, and deploying distributed vector databases (e.g., Milvus or Pinecone) to scale recommendation candidate pools across millions of enterprise requisitions.

ACKNOWLEDGMENT

The authors express their sincere gratitude to their project guide, Prof. Ritu Agrawal (Assistant Professor), project mentor, Prof. Ms. Roshni Solanki, and the Department of Computer

Science and Engineering (AIML) at Parul University for providing invaluable guidance, institutional support, computing infrastructure, and academic supervision throughout the design and evaluation of the AI-ATS platform.

REFERENCES

- [1] J. Chamorro-Premuzic, D. Winsborough, R. Sherman, and T. Hogan, "New talent signals: Shiny new objects or a revolution in talent identification?," *Industrial and Organizational Psychology*, vol. 9, no. 3, pp. 621–640, 2016.
- [2] The Ladders, "Eye-Tracking Study: How Recruiters Really Review Resumes," Tech. Rep., Ladders Inc., New York, NY, 2018.
- [3] P. Cappelli, "Making the approach to talent acquisition work," *Harvard Business Review*, vol. 97, no. 3, pp. 48–59, 2019.
- [4] M. R. Frank, D. Autor, J. E. Bessen, et al., "Toward understanding the impact of artificial intelligence on labor," *Proceedings of the National Academy of Sciences*, vol. 116, no. 14, pp. 6531–6539, 2019.
- [5] E. S. R. M. F. Silva, P. Cortez, and C. E. A. F. Santos, "Using text mining and machine learning to analyze job vacancies and resumes," *Expert Systems with Applications*, vol. 182, p. 115206, 2021.
- [6] R. C. B. C. S. S. Roy, P. K. Singh, and M. D. Mahmud, "Automated Resume Screening System Using Natural Language Processing and Machine Learning," in *Proc. IEEE Int. Conf. on Computing, Communication and Networking Technologies (ICCCNT)*, 2020, pp. 1–6.
- [7] J. H. Park and S. C. Kim, "Spatial document parsing and layout analysis for semi-structured professional profiles," *IEEE Access*, vol. 9, pp. 142105–142116, 2021.
- [8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT*, Minneapolis, MN, 2019, pp. 4171–4186.
- [9] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, 1988.
- [10] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.
- [11] A. Singhal, "Modern information retrieval: A brief overview," *IEEE Data Engineering Bulletin*, vol. 24, no. 4, pp. 35–43, 2001.
- [12] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proc. EMNLP-IJCNLP*, Hong Kong, China, 2019, pp. 3982–3992.
- [13] C. Qin, H. Zhu, T. Xu, C. Shen, E. Chen, and H. Xiong, "Enhancing Person-Job Fit for a Better Career: A Sequential Information Design," *ACM Trans. on Information Systems (TOIS)*, vol. 38, no. 2, pp. 1–28, 2020.
- [14] K. D. Hmieleski and J. C. Carr, "The impact of Applicant Tracking Systems on applicant perceptions and candidate withdrawal," *Journal of Business and Psychology*, vol. 35, no. 4, pp. 511–527, 2020.
- [15] S. A. Hemamali and H. D. Bandara, "AI-driven talent acquisition: Transparent resume evaluation with explainable feedback," in *Proc. IEEE Int. Conf. on Industrial and Information Systems (ICIIS)*, 2022, pp. 301–306.
- [16] P. Tiwary, "FastAPI: Modern, High-Performance Web Framework for Python," *Python Software Foundation*, 2023. [Online]. Available: <https://fastapi.tiangolo.com/>
- [17] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [18] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*. O'Reilly Media, Inc., 2009.