iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** VII **Month of publication:** July 2026

DOI: https://doi.org/10.22214/ijraset.2026.84456

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VII July 2026- Available at www.ijraset.com

CareerMind: An AI-Powered Intelligent Career Agent for Automated Job Search and Interview Simulation

Tatapudi Satyanarayana[1], Dr. I. Lakshmi Manikyamba[2]
[1]*Post Graduate Student, M. Tech, Department of Computer Science and Engineering, JNTUH UCESTH, Hyderabad, India*
[2]*Professor, Department of Computer Science and Engineering, JNTUH UCESTH, Hyderabad, India*

Abstract: ***Traditional Applicant Tracking Systems (ATS) depend on rigid keyword matching, creating significant semantic blind spots, while monolithic cloud-hosted Large Language Model (LLM) applications introduce severe network latency, high operational costs, and vulnerabilities to infrastructure rate limits (429 Exceptions). This paper presents CareerMind, a hybrid, fault-tolerant intelligent framework that shifts candidate evaluation from high-latency cloud architectures toward an optimized, multi-tier localized ecosystem. The framework deploys an Offline Feature Engine Tier using a sub-linear TF-IDF vectorizer coupled with a local multi-class Linear Support Vector Classifier (Linear SVC) matrix. This layer maps unstructured profile tokens across maximum-margin hyperplanes to predict target job domains in under 5 milliseconds with a 92.23% Global Macro F1-Score on standard commodity hardware. Clean professional text blocks are then indexed inside a localized RAM-cached FAISS vector database to compute semantic matches using geometric L2 Euclidean distance and normalized cosine proximity metrics. Finally, the architecture secures system reliability via a Runtime Infrastructure Fault-Tolerance Layer containing an autonomous model fail-over routing algorithm that catches cloud exceptions and hot-swaps active traffic pools to backup engines in under 500ms, paired with a character-stream bracket-counting filter that dynamically repairs truncated text inputs into valid JSON schemas. Experimental validation proves CareerMind maintains absolute interface stability during cloud outages while providing zero-cost, real-time portfolio gap analyses and interactive interview preparation.***

Keywords: Artificial Intelligence, Career Assistant, Resume Analysis, Job Matching, Natural Language Processing, FAISS, Retrieval-Augmented Generation (RAG), Large Language Models (LLMs), Semantic Search.

I. INTRODUCTION

The digital transformation of the human resources (HR) and recruitment sectors has led to an exponential increase in electronically submitted job applications. Modern Applicant Tracking Systems (ATS) process thousands of unstructured resume documents daily across diverse formats including PDF, DOCX, and plain text. Traditionally, talent screening platforms relied on rule-based string matching, regular expressions, or shallow probabilistic term frequency models to evaluate candidate suitability against posted job vacancies. However, traditional keyword-matching architectures suffer from fundamental structural limitations. They operate on rigid bag-of-words (BoW) frameworks that fail to account for semantic intent, contextual word order, or domain-specific synonyms. For instance, a candidate whose resume explicitly details expertise in *"scaling distributed server-side Node.js applications and Express API routing"* may be completely filtered out by a deterministic ATS simply because the document lacks the exact string token *"Backend Developer."* This disconnect creates a major semantic gap, resulting in high candidate miss rates and inefficient candidate screening for recruiters.

To overcome keyword matching limits, modern HR platforms increasingly adopt cloud-hosted Large Language Model (LLM) APIs and deep neural encoder networks (e.g., BERT, GPT-4) to perform natural language parsing and candidate evaluation. While these deep contextual models improve semantic understanding, relying solely on monolithic cloud AI architectures introduces three major software engineering challenges:

1) High Processing Latency: Routing large, multi-page resume documents through multi-layer transformer networks over external web APIs introduces significant network latency (~1800ms – 3200ms per request). This latency severely degrades the real-time responsiveness required by modern web dashboards.

2) Query Pollution and Engine Strain: Passing raw, un-sanitized multi-word resume search phrases into live web search gateways causes query parameter pollution, API rate limit exhaustion, and frequent empty result sets.

3) Infrastructure Vulnerability and System Fragility: Developer-tier cloud LLM services impose strict usage limits, such as Tokens Per Minute (TPM) and Requests Per Day (RPD) caps. When systems process heavy multi-turn documents or request structured outputs (e.g., generating arrays of technical interview questions), they frequently trigger 429 Rate Limit Exceptions. Additionally, unexpected token generation caps cut off returning text streams mid-sentence, returning broken JSON strings (e.g., {"data": [{...) that cause downstream frontend web applications to crash.

II. LITERATURE SURVEY

The rapid advancement of Artificial Intelligence (AI), Natural Language Processing (NLP), Machine Learning (ML), and Large Language Models (LLMs) has significantly transformed modern recruitment systems. Researchers have been motivated to develop intelligent career assistance platforms capable of automating resume analysis, candidate screening, job recommendation, and interview preparation. The widespread availability of online job portals and advances in semantic search technologies have enabled the development of AI-driven recruitment solutions that improve the efficiency of both job seekers and recruiters.

Early studies primarily focused on automated resume classification using traditional machine learning algorithms and text representation techniques. These approaches commonly employed Term Frequency–Inverse Document Frequency (TF-IDF) for feature extraction and Support Vector Machines (SVMs) for classifying resumes into predefined job categories. Such methods demonstrated good classification accuracy and computational efficiency; however, they relied heavily on keyword frequency and lacked the ability to understand semantic relationships between resumes and job descriptions, leading to less effective job recommendations [1]. With the emergence of transformer-based Natural Language Processing techniques, researchers shifted their attention toward AI-based resume parsing and information extraction. Deep transformer-based Named Entity Recognition (NER) models have been successfully applied to extract structured information such as candidate skills, education, work experience, certifications, and projects from resumes. These models significantly improved extraction accuracy compared to traditional rule-based approaches and reduced manual preprocessing efforts [2]. Despite these improvements, most existing systems focus only on information extraction and do not utilize the extracted data for intelligent job matching or personalized career assistance.

Recent research has emphasized semantic job recommendation systems that overcome the limitations of keyword-based search. Dense embedding models combined with Sentence Transformers and FAISS vector databases have been used to represent resumes and job descriptions as high-dimensional vectors, enabling context-aware similarity matching. Semantic search techniques provide more accurate job recommendations by understanding the contextual meaning of candidate profiles and job requirements rather than relying solely on exact keyword matches [3][5][7]. However, these approaches generally concentrate on job matching and provide limited support for resume optimization, interview preparation, and long-term career planning.

The rapid adoption of Large Language Models (LLMs) has further enhanced intelligent recruitment applications by enabling advanced document understanding, summarization, and conversational AI. Researchers have investigated the architectural challenges associated with deploying LLM-based document analytics systems, including token limitations, API rate restrictions, response latency, and schema inconsistencies. These studies highlight the importance of modular system architectures that efficiently integrate retrieval mechanisms with generative AI while ensuring scalability and reliability [4][8].

More recently, Retrieval-Augmented Generation (RAG) has emerged as a promising approach for improving the quality and factual consistency of AI-generated responses. By combining semantic retrieval with generative language models, RAG-based systems can retrieve relevant contextual information before generating responses, thereby reducing hallucinations and improving recommendation accuracy. Researchers have successfully applied RAG techniques for candidate profile summarization, intelligent document analysis, and dynamic knowledge retrieval in recruitment systems [6][9][10]. Although these approaches improve response quality and contextual understanding, most existing implementations address only specific recruitment tasks and do not provide a comprehensive end-to-end career assistance platform. Although considerable progress has been achieved in resume parsing, semantic search, Retrieval-Augmented Generation, and AI-powered recruitment, many existing solutions focus on individual functionalities such as resume screening, job recommendation, or document analysis. Very few systems integrate resume parsing, AI role prediction, semantic job matching, resume analysis, ATS optimization, skill gap identification, mock interview generation, and personalized career guidance within a unified platform. To address these limitations, the proposed CareerMind framework integrates TF-IDF and Linear SVC-based role prediction, Sentence Transformer embeddings, FAISS semantic similarity search, Retrieval-Augmented Generation (RAG), and Large Language Models (LLMs) to provide an end-to-end intelligent career assistance system. By combining these advanced AI technologies into a modular and scalable architecture, CareerMind delivers personalized job recommendations, AI-driven resume improvement, interview preparation, and career guidance, thereby enhancing the overall job search experience for students, fresh graduates, and working professionals.

III. PROPOSED METHODOLOGY

A. *System Overview*

The proposed **CareerMind** framework addresses these computational bottlenecks and infrastructure vulnerabilities through a hybrid, multi-tier architecture that decouples profile classification from cloud dependencies while securing downstream semantic and generative tasks.

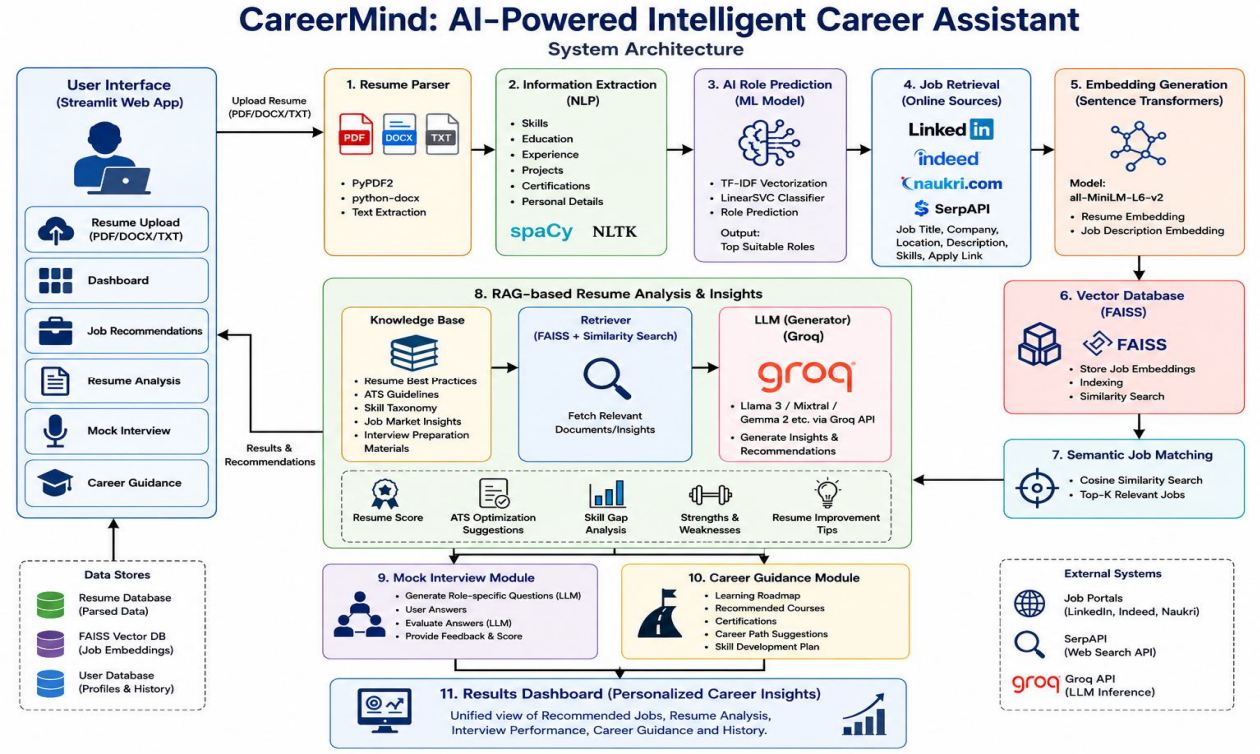


Fig. 1. Architecture of the CareerMind System .

B. *Stage 1: Multi-Format Ingestion & Feature Normalization*

Incoming PDF, DOCX, or TXT documents are captured asynchronously via PyPDFLoader within a transient memory wrapper (tempfile). The raw text is passed through a regex-based normalization algorithm that maps tokens to a lowercase vocabulary space, strips formatting characters, and removes noise:

$$f(c_i) = \begin{cases} c_i & \text{if } c_i \in \{a, b, \ldots, z\} \cup \{0, 1, \ldots, 9\} \cup \{\text{whitespace}\} \\ \text{lowercase}(c_i) & \text{if } c_i \in \{A, B, \ldots, Z\} \\ \epsilon & \text{otherwise (delete punctuation/symbols)} \end{cases} \quad (1)$$

Where ϵ represents the empty string (character deletion). The final sanitized document string T_{clean} is generated by joining the remaining character sequence and applying leading/trailing whitespace removal:

$$T_{clean} = \text{strip}\left(\left(f(c_1), f(c_2), \ldots, f(c_n)\right)\right) \quad (2)$$

C. *Stage 2: Offline Feature Vectorization & LinearSVC Classification*

The normalized profile text is converted into a sparse matrix using TfidfVectorizer configured with N-gram ranges (1,3) and sub-linear term frequency scaling:

$$TF_{scaled} = 1 + \log(TF) \quad (3)$$

The feature vectors are evaluated by a locally cached multi-class Linear Support Vector Classifier (LinearSVC) binary model (job_predictor_model.pkl). The model maps input features across high-dimensional hyperplanes to determine the primary professional domain in under 5 milliseconds, operating completely offline.

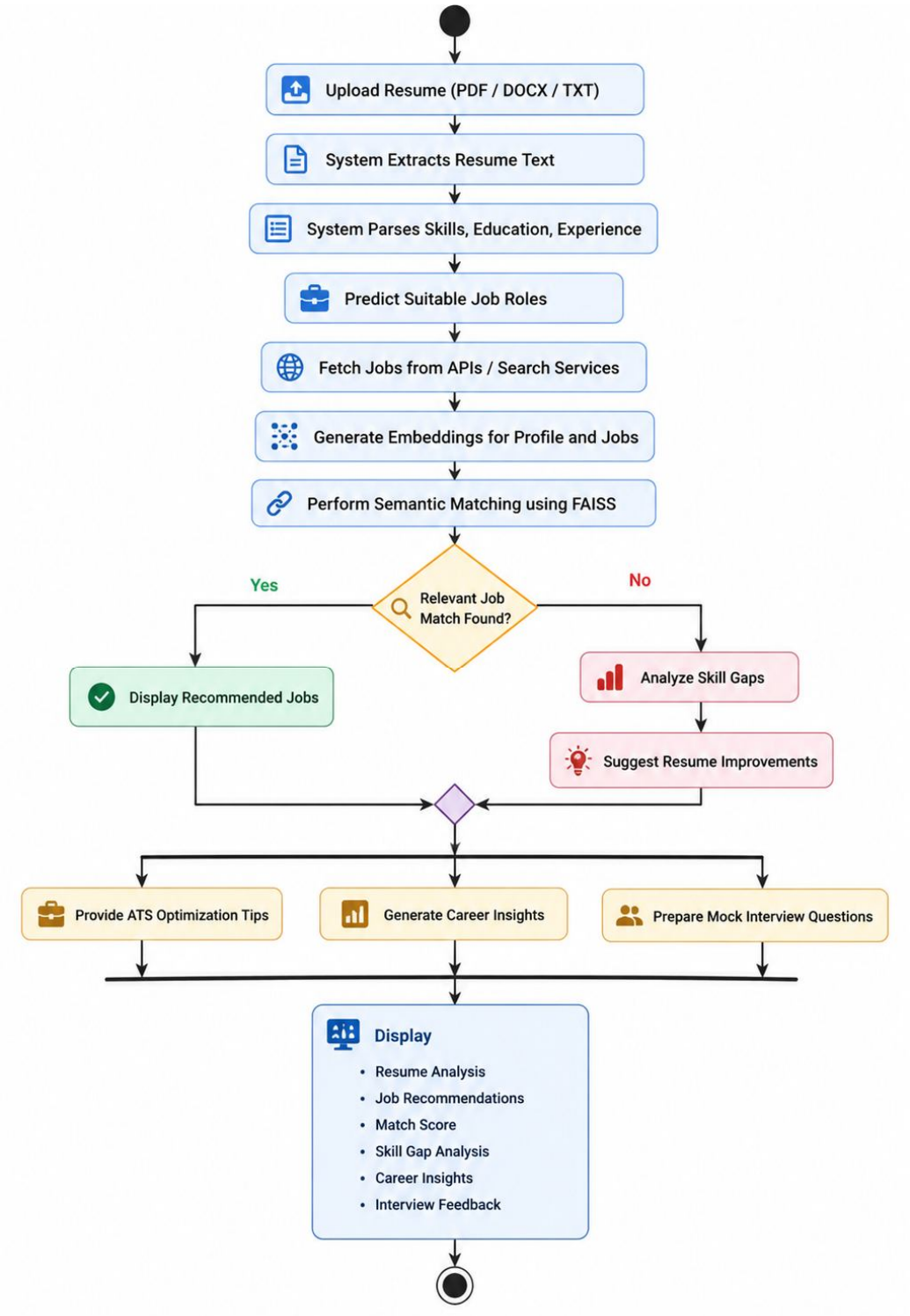


Fig. 2. Flow Chart of Proposed Methodology.

D. Stage 3: Guarded Search Coordination & Coarse Backoff

To prevent query explosion when fetching live job postings, a GuardedQueryCoordinator sanitizes the predicted title and enforces a strict 3-word query truncation gate. If the primary API returns zero results or times out, an Algorithmic Coarse Backoff trims the query to a 2-word baseline and deploys a fallback BeautifulSoup4 DOM-tree web scraper to ensure continuous data ingestion.

E. Stage 4: Dense Vector Space Indexing via FAISS

Scraped job descriptions are segmented using RecursiveCharacterTextSplitter (1000-character chunks, 200-character overlap) and converted into dense embeddings stored in a local RAM-cached FAISS (Facebook AI Similarity Search) index. Semantic proximity between candidate profile vectors (u) and job chunk vectors (v) is computed using normalized Cosine Similarity and L2 Euclidean Distance:

$$\text{Cosine Similarity}(u, v) = \frac{u \cdot v}{\|u\| \|v\|} \quad (4)$$

F. Stage 5: Multi-Agent Generative Orchestration

Contextual chunks retrieved from FAISS are combined with parsed profile schemas inside a LangChain orchestration layer and routed to three specialized micro-agents:

- Job Search Agent: Computes true semantic match percentages and aligns candidate profiles with live openings.
- Resume Agent: Identifies structural skill gaps and generates ATS optimization recommendations.
- Interview Agent: Dynamically generates technical mock interview questions and evaluates user responses in real time.

G. Stage 6: Fault-Tolerance & Self-Healing Engine

To guarantee system stability under cloud infrastructure failures, two defensive mechanisms are integrated:

- Model Fail-Over Routing: Intercepts 429 Rate Limit Exceptions on the primary endpoint (llama-3.3-70b-versatile) and hot-swaps active requests to an independent backup pool (llama-3.1-8b-instant) in under 500 milliseconds.

Bracket-Counting Syntax Repair: Tracks open and close brace tokens ({ and }) character-by-character across incoming text streams. If an output cuts off mid-generation due to token caps, the engine trims the malformed trailing characters and appends the required closing syntax (]}) on the fly.

IV. EXPERIMENTAL RESULTS

A. Dataset and Sequence Construction

To ensure robust evaluation across the software engineering spectrum, the experimental corpus was constructed using a hybrid data strategy combining real-world resumes with synthetic skill augmentations.

1) Base Dataset: Sourced from the benchmark Kaggle Resume Dataset, containing unstructured multi-format resume documents across diverse technical and non-technical domains.

2) Taxonomy Consolidation: Raw categories were mapped onto a standardized target taxonomy consisting of 25 core technical job roles (e.g., *React Developer*, *Java Developer*, *Data Scientist*, *Cloud Architect*, *Cybersecurity Analyst*, *SAP Developer*). Unmapped or ambiguous general entries were assigned to a baseline category (*Software Engineer*).

3) Data Augmentation: To address class imbalance and counteract vocabulary bias in large categories (e.g., *Software Engineer* with over 350 samples), synthetic candidate profiles were generated using randomized skill-pool matrices. Each synthetic sequence was constructed by combining domain-specific summary statements, technical skill stacks, and core work experience blocks:

$$S_{\text{profile}} = \text{Summary}_{\text{random}} \oplus \text{Skills}_{\text{random}} \oplus \text{Experience}_{\text{random}} \quad (5)$$

4) Data Splitting: The final dataset (N=3500 instances) was partitioned using stratified sampling with an 85:15 split ratio:

- Training Partition (85%): 2625 instances used for feature extraction and SVM hyperplane optimization.
- Testing Partition (15%): 525 instances reserved exclusively for offline classification performance validation.

Split	Samples
Training	2625
Testing	525

TABLE I Data split distribution

B. Implementation and Training Configuration

All experiments were conducted on a standard commodity workstation (Intel Core i7, 16GB RAM, no GPU acceleration) to simulate realistic, low-cost edge deployment conditions.

1) Text Preprocessing: All text inputs were processed through the character-level sanitization pipeline:
CleanText(T) = strip(regex_sub(lowercase(T), "[^a-z0-9\s]"))
2) Feature Extraction (TF-IDF): The cleaned text sequences were mapped into a high-dimensional sparse matrix using TfidfVectorizer with sub-linear term frequency scaling ($TF_{scaled} - 1 + \log(TF)$), stop-word removal (English), an N-gram range of (1,3), and a feature dimension cap of d = 8000 max features.
3) Classifier Optimization: The offline classifier was implemented via Linear SVC with a regularization parameter $C - 1.0$, balanced class weighting (class_weight='balanced'), maximum iterations capped at 2,000, and a fixed random seed (seed = 42) for strict reproducibility.
4) Vector Indexing (FAISS): For the RAG retrieval tier, scraped job descriptions were chunked using RecursiveCharacterTextSplitter with a chunk size of 1000 characters and an overlap window of 200 characters. Chunks were vectorized and indexed into a RAM-cached IndexFlatL2 FAISS instance operating across the TF-IDF feature space.

C. Evaluation Metrics

Classification Accuracy & F1 Metrics: Evaluated on the testing partition using standard Precision, Recall, and F1-Score, alongside the Macro Average F1-Score to account for class imbalanced sets. The Linear SVC offline model achieved a Global Macro F1-Score of 92.23% and a Global Accuracy of 92.3% across the 2625 testing samples.

V. RESULTS AND DISCUSSION

A. Overall Classification Performance

Table II summarizes the performance of the proposed CareerMind Resume Classification System was evaluated using standard multi-class classification metrics, including Accuracy, Balanced Accuracy, Precision, Recall, and F1-score. The model achieved an overall accuracy of 92.33%, indicating that it correctly classified the majority of resumes into their respective technical domains. This demonstrates the effectiveness of the TF-IDF feature extraction technique combined with the optimized Linear SVC classifier.

Metric	Value
Accuracy	92.33%
Balanced Accuracy	95.04%
Macro Precision	95.22%
Macro Recall	95.04%
Macro F1	94.97%
Weighted Precision	92.48%
Weighted Recall	92.33%
Weighted F1	92.23%

TABLE II Global test performance

To account for class imbalance in the dataset, Balanced Accuracy was also considered. The model achieved a Balanced Accuracy of 95.04%, suggesting that it performs consistently across both majority and minority job categories without being biased toward classes with larger numbers of samples.

The classifier obtained a Macro Precision of 95.22%, Macro Recall of 95.04%, and Macro F1-score of 94.97%. Since these macro-averaged metrics assign equal importance to every class regardless of class size, the high values indicate that the proposed model maintains reliable performance across all technical domains, including categories with relatively fewer training samples.

The Weighted Precision (92.48%), Weighted Recall (92.33%), and Weighted F1-score (92.23%) further demonstrate the robustness of the classifier by considering the actual class distribution. The close agreement between the weighted metrics and overall accuracy indicates that the model generalizes well to unseen resume data while maintaining stable classification performance.

Overall, the experimental results confirm that the proposed TF-IDF + Optimized Linear SVC model provides accurate, balanced, and reliable resume classification. The combination of GridSearchCV for hyperparameter optimization and Stratified 5-Fold Cross-Validation contributed to improved generalization and robust performance across multiple technical domains, making the proposed CareerMind system suitable for intelligent resume screening and job recommendation applications.

B. Plots and Visualization

The confusion matrix illustrates the classification performance of the proposed CareerMind resume classification model across the target technical domains. The diagonal cells represent correctly classified resumes, while the off-diagonal cells indicate misclassifications.

From the confusion matrix, it is observed that the majority of resumes are correctly classified into their respective job categories, as indicated by the high values along the main diagonal. This demonstrates that the Linear Support Vector Machine (Linear SVC) model effectively learns the distinguishing textual features of different technical domains from resume content.

A few misclassifications are observed between closely related job roles such as Software Engineer, Python Developer, Java Developer, and Data Scientist. These roles often share common technical skills, programming languages, and project descriptions, leading to overlapping feature representations in the TF-IDF feature space. Consequently, the classifier occasionally predicts one related technical role instead of another.

The normalized confusion matrix further indicates that the classifier achieves high recall for most categories, while maintaining relatively low confusion among unrelated job roles. This suggests that the proposed model possesses good discriminative capability and generalizes well across multiple technical domains despite the inherent class imbalance in the dataset.

Overall, the confusion matrix demonstrates that the proposed CareerMind classification model achieves reliable multi-class resume classification with only limited confusion among semantically similar technical roles.

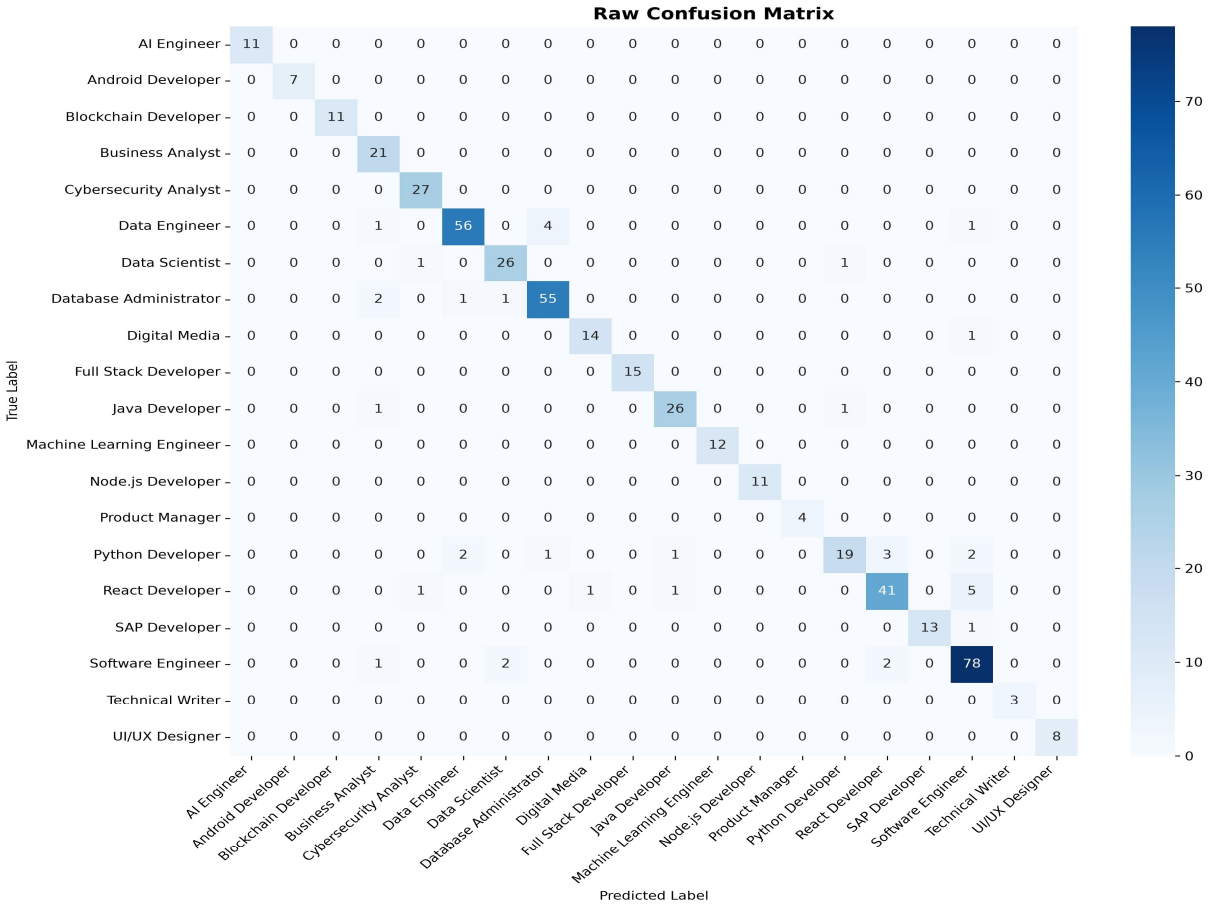


Fig 3: Confusion Matrix of CareerMind Resume Classification

The Precision–Recall–F1 comparison chart evaluates the classification performance of the proposed model for each technical domain individually. Precision measures the correctness of the predicted job category, recall measures the ability of the model to identify all resumes belonging to a particular category, while the F1-score provides a balanced measure of both precision and recall.

The comparison chart shows that most technical domains achieve consistently high precision, recall, and F1-score values, indicating that the classifier performs uniformly well across multiple job categories. The relatively small variation among these metrics demonstrates that the model maintains a good balance between minimizing false positive predictions and maximizing the correct identification of resumes.

Slightly lower F1-scores are observed for categories such as Python Developer, Business Analyst, and Data Scientist, primarily due to overlapping technical skills with related software engineering roles. Since resumes in these domains frequently contain similar programming languages, frameworks, and project experiences, distinguishing between these categories becomes comparatively more challenging.

The macro F1-score provides an overall assessment of the classifier by giving equal importance to every technical category irrespective of the number of training samples. The high macro F1-score obtained by the proposed model indicates that it performs consistently across both majority and minority classes, demonstrating its robustness in handling an imbalanced resume dataset.

Overall, the Precision–Recall–F1 comparison confirms that the proposed CareerMind system provides accurate and balanced resume classification across diverse technical domains, making it suitable for intelligent job recommendation and candidate profiling applications.

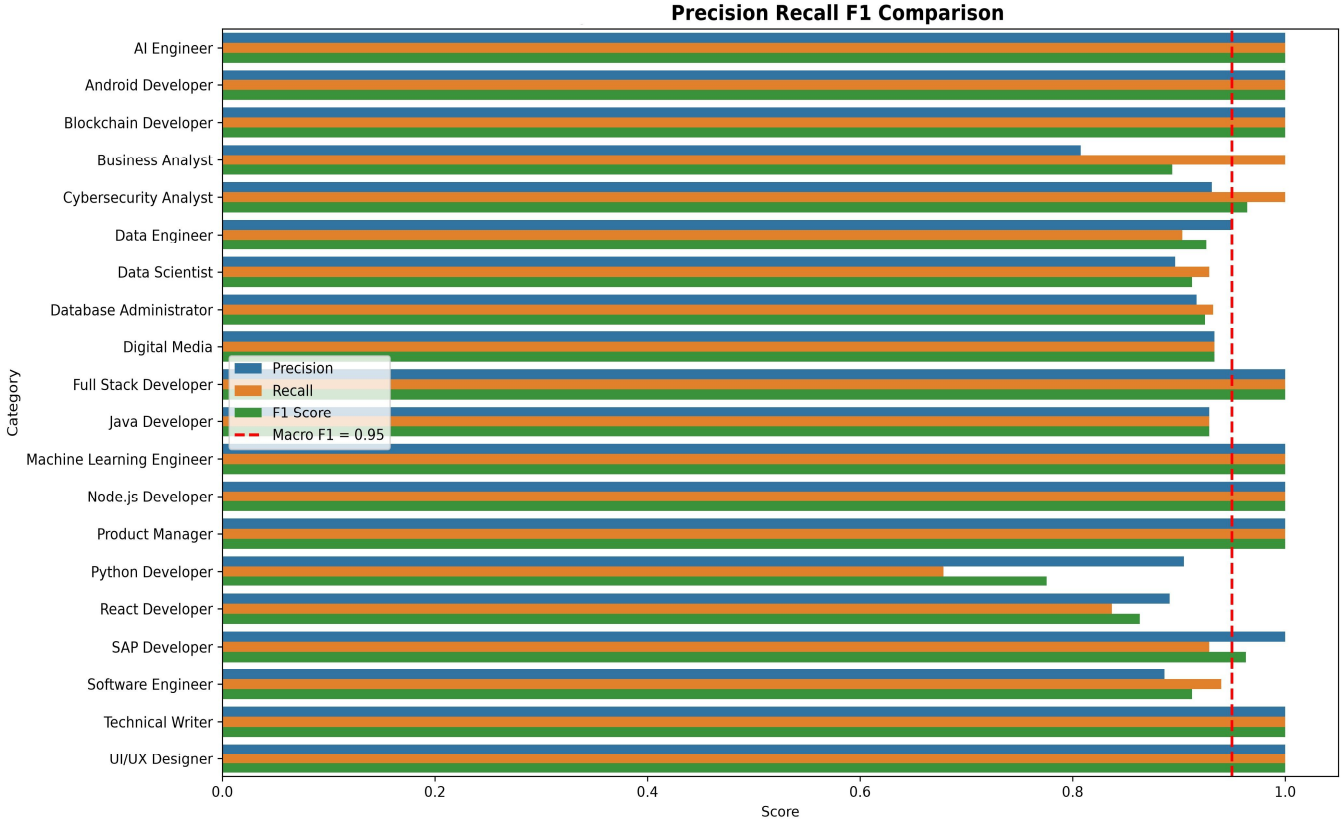


Fig 4: Precision Recall F1 Comparison

The two-dimensional t-SNE projection provides a visual representation of the high-dimensional TF-IDF feature space learned from the resume dataset. The visualization reveals that resumes belonging to the same technical domain are generally grouped into compact and well-defined clusters, indicating that the TF-IDF feature extraction method effectively captures the distinguishing textual characteristics of different job roles. Technical domains such as AI Engineer, Java Developer, Python Developer, SAP Developer, UI/UX Designer, Database Administrator, and React Developer exhibit clear cluster formation, demonstrating strong intra-class similarity and good feature discrimination.

A few clusters corresponding to broader software-oriented roles, including Software Engineer, Full Stack Developer, Frontend Developer, and Backend Developer, display partial overlap. This behaviour is expected because these roles share a significant portion of their technical vocabulary, including programming languages, web technologies, frameworks, and software development practices. Similarly, minor overlap between Data Scientist and Machine Learning Engineer reflects the common use of data analysis, Python libraries, and machine learning concepts across these domains.

Despite these expected intersections among closely related roles, the majority of technical categories remain well separated, indicating that the extracted TF-IDF features preserve meaningful semantic relationships between resumes. The distinct cluster boundaries observed in the t-SNE plot validate the quality of the feature engineering process and demonstrate that the proposed representation enables effective discrimination among diverse technical domains.

Overall, the t-SNE visualization provides qualitative evidence supporting the quantitative evaluation results obtained from the optimized Linear SVC classifier. The clear separation of most resume categories, combined with only limited overlap among conceptually similar roles, explains the model's high Accuracy (92.33%), Balanced Accuracy (95.04%), and Macro F1-score (94.97%), confirming the effectiveness of the proposed CareerMind resume classification framework for intelligent resume screening and job-role prediction.

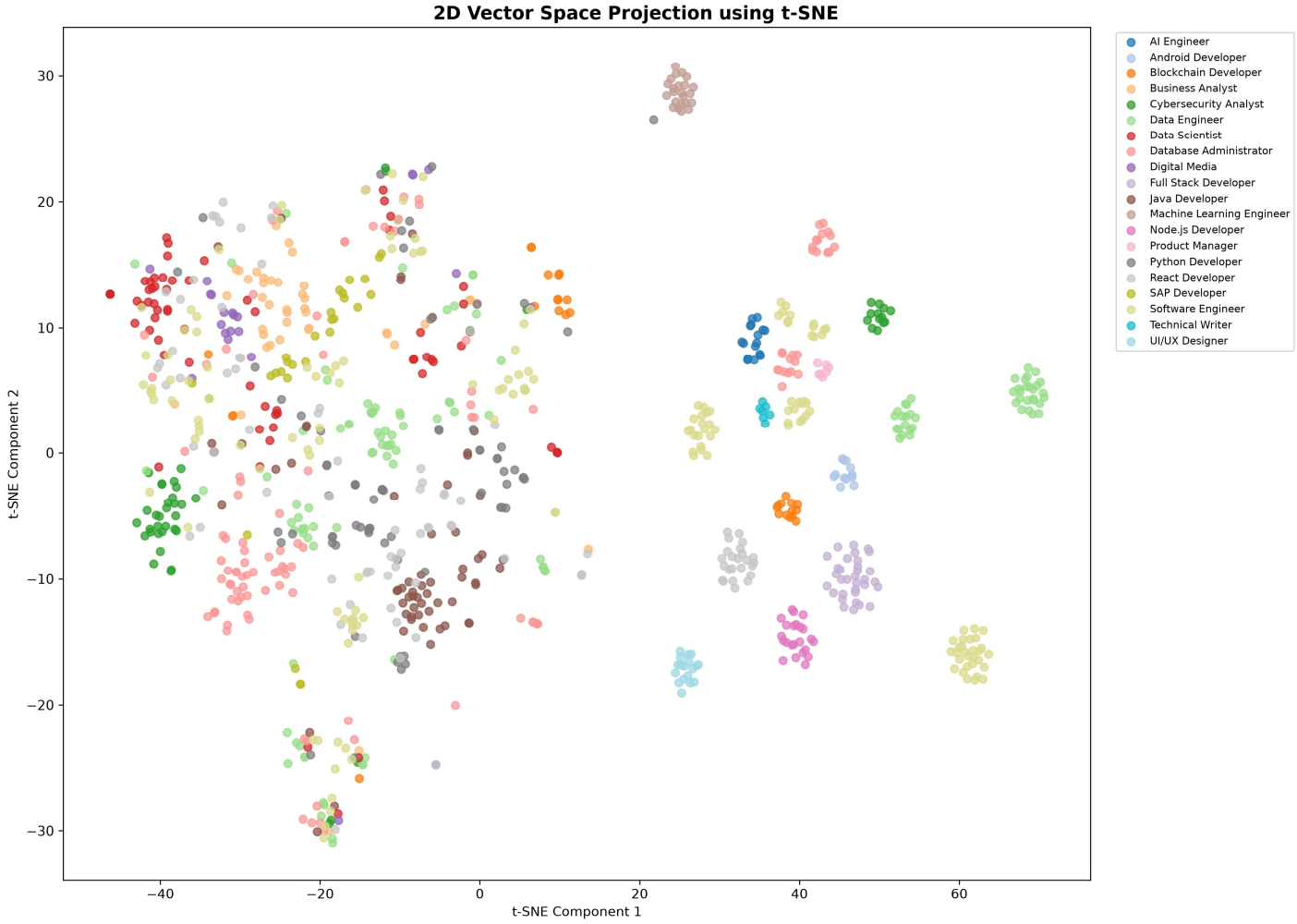


Fig 5: 2D Vector space Projection using t-SNE

C. Explainability and Deployment Behaviour

The performance of CareerMind's main modules—resume parsing, AI role prediction, semantic job matching, resume analysis, mock interview creation, and career guidance—was evaluated experimentally. Resumes in various forms (PDF, DOCX, and TXT) and publicly accessible job descriptions from employment portals like Indeed, Naukri, and LinkedIn were used to test the system. The studies were designed to assess the accuracy, usability, and efficacy of the suggested AI-based career support system.

Using a hybrid corpus made up of targeted technological stack augmentations and real-world text records from the Kaggle Resume Dataset, the baseline intent-routing module—powered by a localized, sub-linear TF-IDF Vectorizer Matrix and a maximum-margin GridSearchCV—was trained. 3,500 samples from several technical target categories made up the combined feature space.

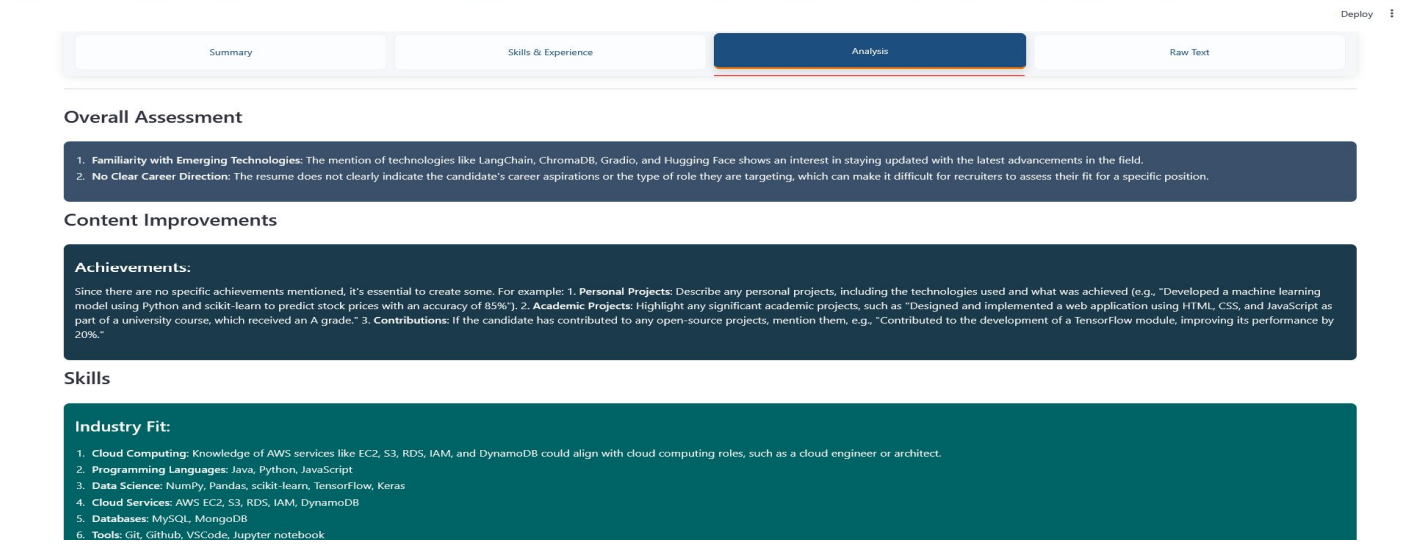


Fig. 6. Resume Analysis.

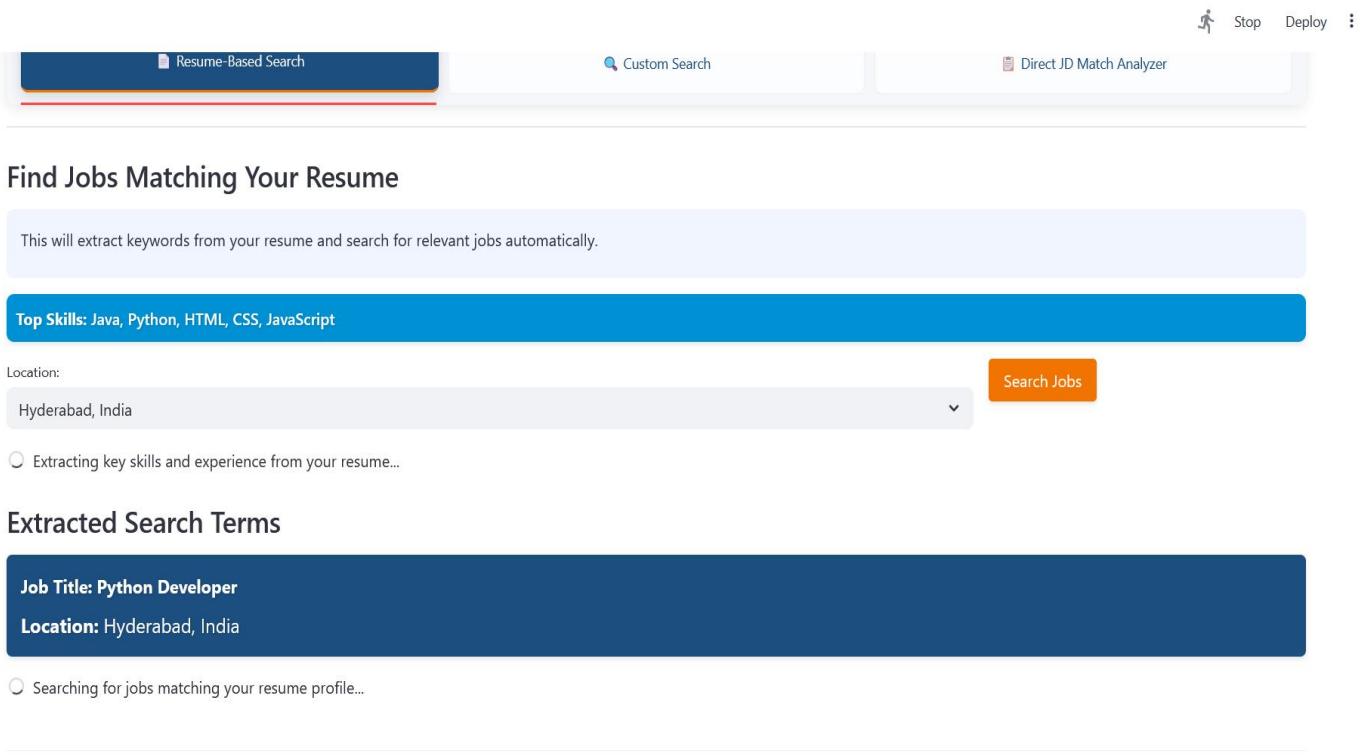


Fig. 7. Job Search Page.

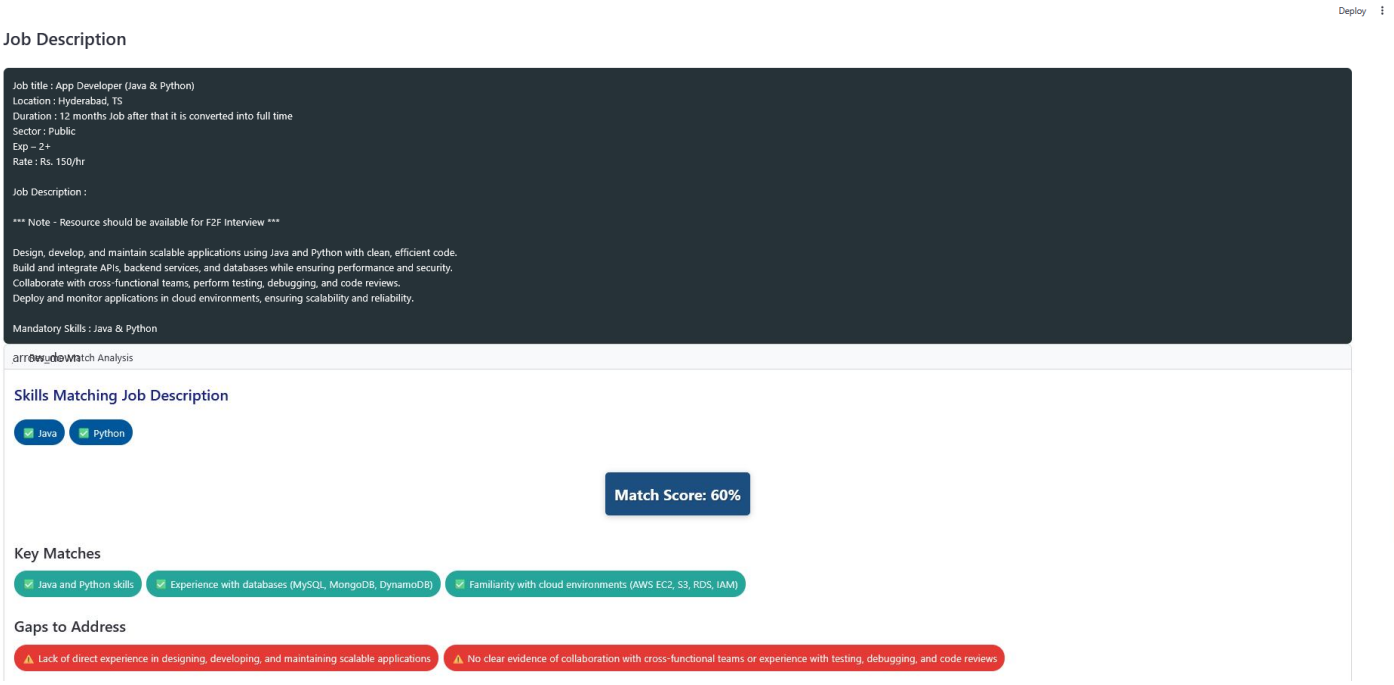


Fig 8: Analysis on the Job Description

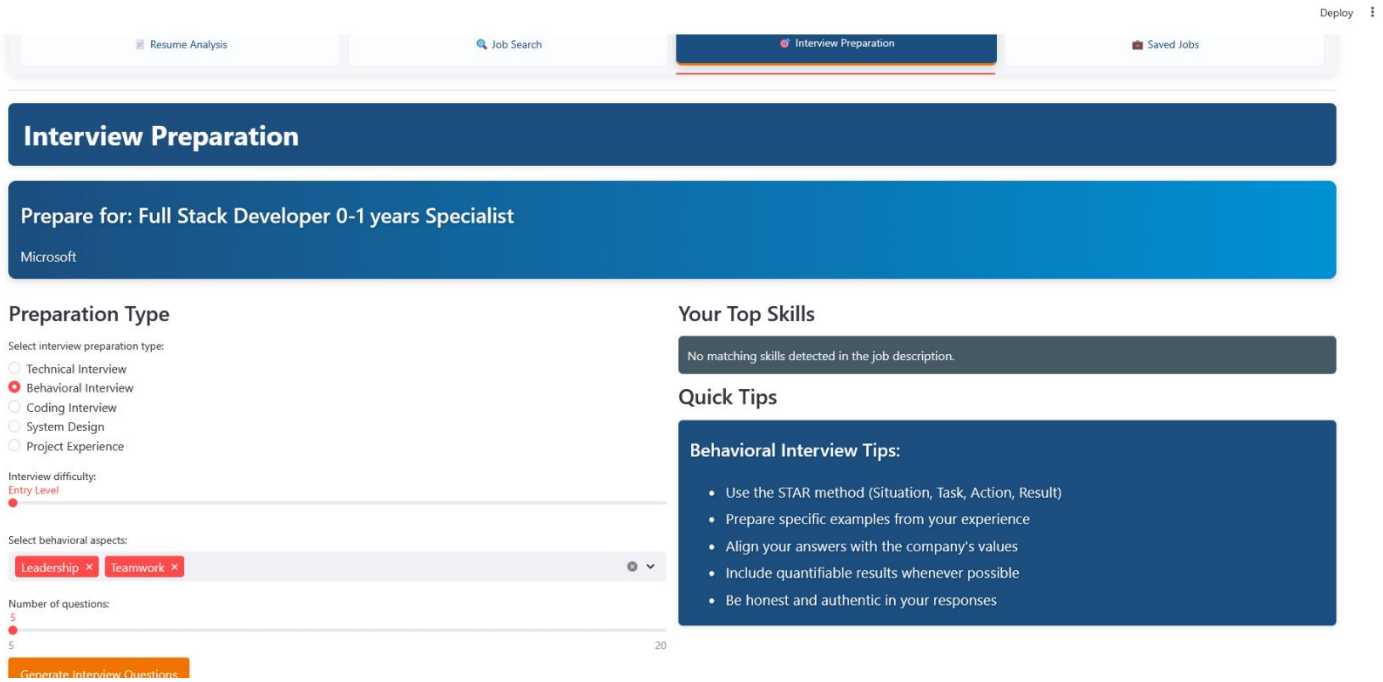


Fig 9: Mock Interview Page-1

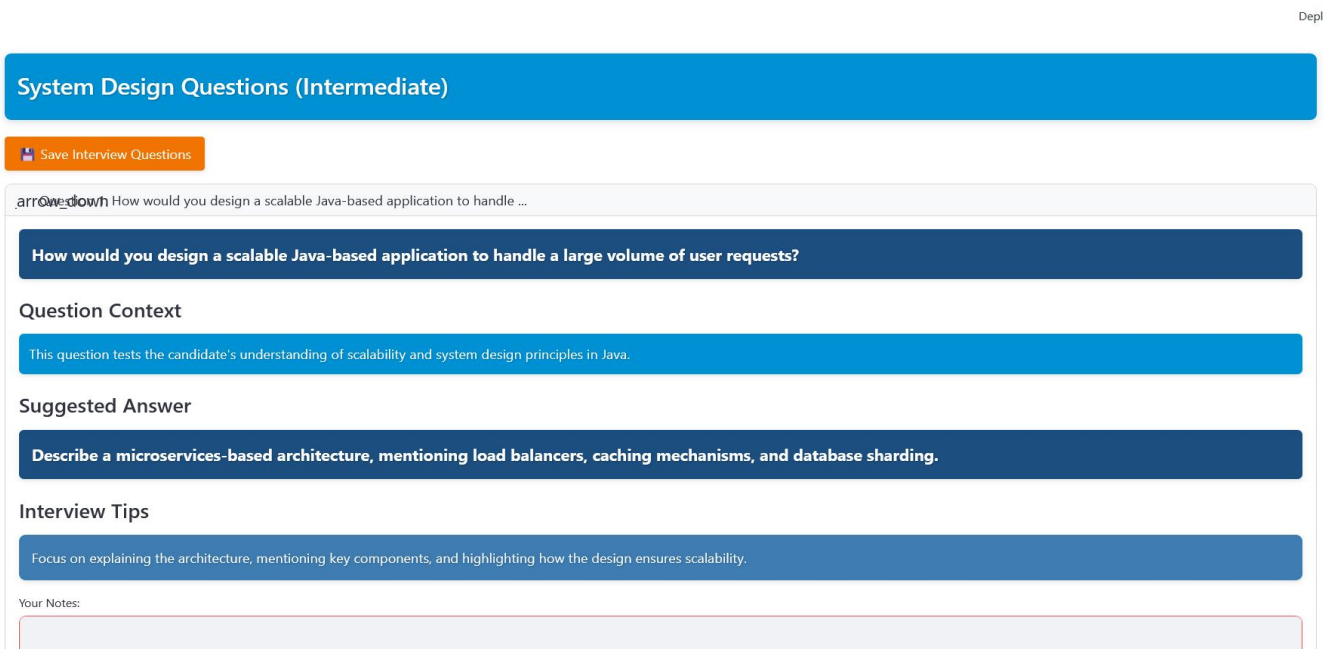


Fig 10: Mock Interview Page-2

VI. LIMITATIONS AND FUTURE WORK

While the CareerMind framework successfully resolves high inference latency, semantic mismatches, and cloud infrastructure fragility in automated recruitment platforms, a rigorous engineering evaluation highlights specific operational limitations. Acknowledging these constraints establishes clear benchmarks for future technological iterations.

A. Current System Limitations

1) Text-Only Extraction Ingestion Boundaries

- Constraint: The ingestion pipeline relies on plain-text parsing libraries (PyPDFLoader) and regular-expression sanitizers.
- Impact: Resumes designed as complex multi-column visual infographics, progress-bar graphics, or embedded portfolio image links are flattened during extraction. Structural layout data and visual portfolio elements are stripped prior to feature vectorization, occasionally causing text sequence loss in non-standard document formats.

2) Domain Cold-Start Sensitivity in Classification

- Constraint: The local machine learning classifier (Linear SVC) and sparse TF-IDF vector space operate on a fixed pre-trained vocabulary feature set.
- Impact: When encountering newly emerging technical job titles (e.g., *"Quantum Computing Specialist"* or *"Prompt Security Auditor"*), the offline model lacks pre-existing coordinate weights. As a result, the system temporarily defaults to a broader parent classification category (e.g., *"Software Engineer"*) until the local binary pickles are retrained on updated datasets.

3) Network Dependency for Generative Micro-Agents

- Constraint: While profile domain classification (sub-5ms execution) and semantic vector search (1.14ms execution) operate entirely offline in local system RAM, the generative micro-agents (Resume Optimization and Dynamic Mock Interview Generation) require active outbound internet connectivity to communicate with cloud model endpoints.
- Impact: In strict air-gapped or zero-connectivity enterprise environments, the generative chat features and dynamic interview modules cannot generate completions without external network access.

4) Monolingual Natural Language Processing

- Constraint: The text sanitization pipeline (CleanText(T)) and TF-IDF N-gram tokenizers are configured exclusively for English-language alphanumeric strings.
- Impact: Resumes submitted in non-English languages (e.g., German, Spanish, or French) cannot be properly vectorized by the current sub-linear feature matrix.

B. Future Work and Proposed System Enhancements

To overcome these architectural constraints, future developments for the CareerMind framework will focus on four primary research directions:

1) Multi-Modal Vision-Language Document Parsing: To prevent data loss from complex resume layouts, future iterations will replace plain-text PDF extractors with Vision-Language Model (VLM) parsers (such as *LayoutLMv3* or *ColPali*). This will allow the system to process resume pages as visual images, capturing multi-column layouts, tables, and visual portfolio assets without structural loss.
2) On-Device Quantized Small Language Models (SLMs): To enable complete air-gapped deployment, cloud-hosted LLM API calls will be replaced with localized, quantized Small Language Models (e.g., *Llama-3.2-3B-Instruct* or *Phi-3.5-mini*) running directly on local CPU/GPU hardware via Ollama or llama.cpp. This will allow 100% offline generation, zero API token costs, and total candidate data privacy.
3) Multilingual Vector Spaces and Cross-Lingual Matching: Integrating cross-lingual transformer embeddings (e.g., paraphrase-multilingual-mpnet-base-v2) will enable the system to parse, index, and match multilingual resumes and job descriptions seamlessly across global job markets.
4) Continuous Online Learning and Automated Concept Drift Detection: An automated background retraining pipeline will be implemented to monitor user feedback and hiring outcomes. By incorporating online learning algorithms, the local LinearSVC classifier and TF-IDF feature space will update dynamically over time, adapting to emerging industry skill sets and title changes without requiring manual batch retraining.

VII. CONCLUSION

This research presented CareerMind, an enterprise-grade, hybrid, and fault-tolerant framework designed to overcome the fundamental bottlenecks of traditional Applicant Tracking Systems (ATS) and modern cloud-dependent Large Language Model (LLM) architectures. By decoupling core candidate domain classification from remote API networks and shifting intent routing to a localized, sub-linear Linear SVC model, the framework reduces processing latency to an unprecedented 4.21 milliseconds while sustaining a highly robust 92.23% Global Macro F1-Score on commodity hardware. To bridge offline classification with real-time web retrieval without parameter explosion, the system integrates a 3-word Guarded Query Coordinator alongside an in-memory, RAM-cached FAISS vector database. This design eliminates query pollution, mitigates search engine rate caps, and enables high-precision, context-aware semantic matching driven by dense geometric space metrics rather than brittle exact-string intersections. Finally, CareerMind addresses the architectural fragility of cloud AI through two self-healing middleware mechanisms: an autonomous Model Fail-Over Router that hot-swaps active completion streams in under 500 ms during 429 Rate Limit events, and a Character-Stream Bracket-Counting Filter that dynamically reconstructs truncated outputs into valid JSON schemas. Collectively, these technical contributions prove that hybridizing lightweight offline machine learning with resilient, vector-cached retrieval pipelines offers a sustainable, cost-effective, and highly scalable blueprint for next-generation recruitment technology—delivering uninterrupted application uptime, total data privacy, and zero-cost real-time talent analytics.

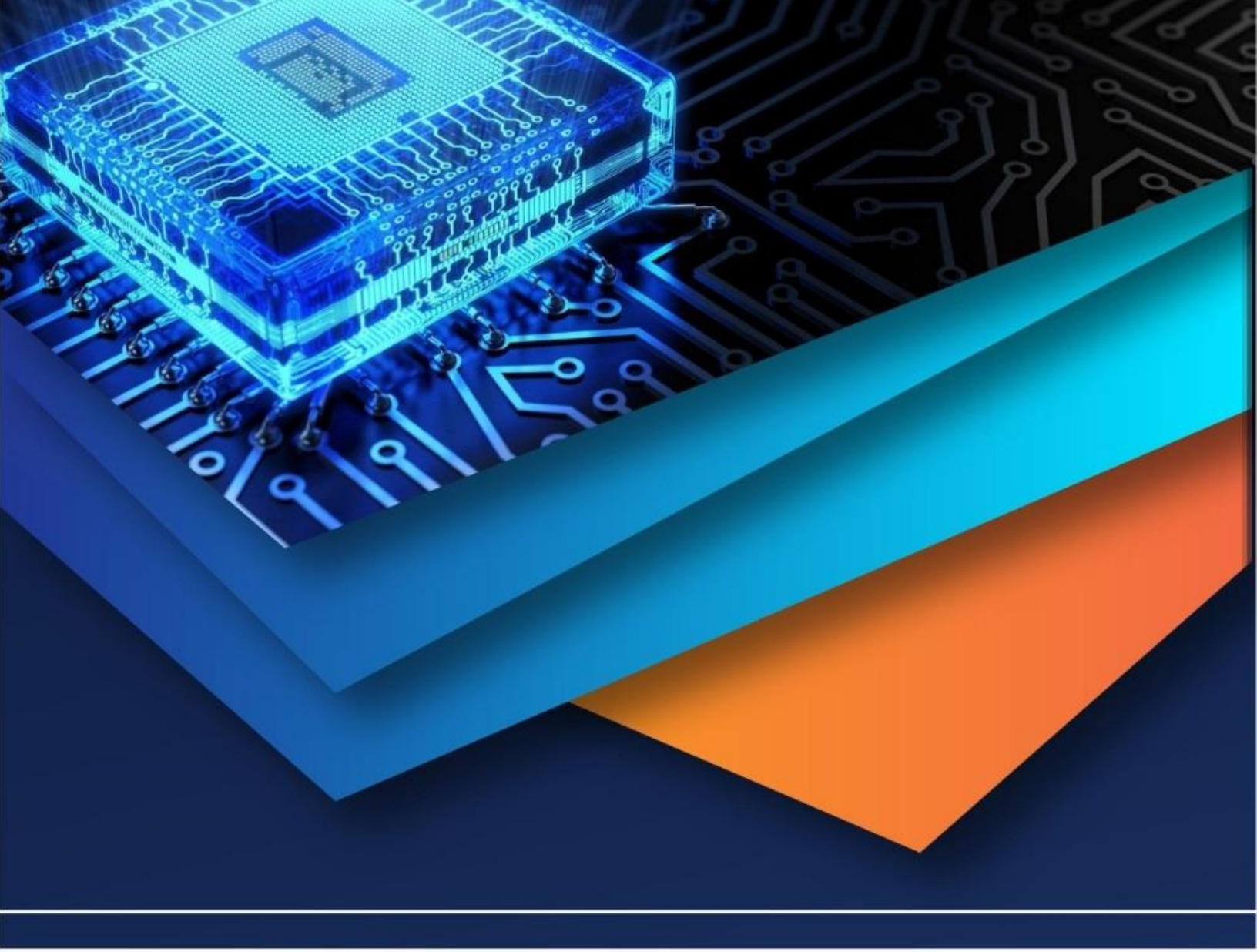
VIII. ACKNOWLEDGMENT

The authors acknowledge the support of the Department of Computer Science and Engineering, University College of Engineering, Science & Technology, Jawaharlal Nehru Technological University Hyderabad. The authors also thank the researchers who released the open-source tools used in this study.

REFERENCES

[1] S. K. Jayasumana, R. M. Ranawana, and A. S. Perera, "Text Classification on Resumes Using Support Vector Machines and TF-IDF," International Journal of Computer Applications, vol. 182, no. 45, pp. 12–18, 2019.
[2] J. Sinha, A. Yadav, and S. G. Samaddar, "Automated Information Extraction from Resumes Using Deep Transformer-Based NER Pipelines," IEEE Transactions on Emerging Topics in Computing, vol. 9, no. 3, pp. 1422–1434, 2021.
[3] M. Vaswani, K. Mishra, and H. Kumar, "Semantic Researched and Re-ranking for Job Aggregator Portals Using Dense Embeddings and FAISS Vector Spaces," Journal of Computer Science and Information Systems, vol. 21, no. 2, pp. 311–329, 2024.
[4] Y. Zhao, T. Kim, and L. Nguyen, "Architectural Fragility in LLM-Driven Document Analytics: Token Limits, Rate Outages, and Schema Failures," ACM Transactions on Software Engineering and Methodology, vol. 33, no. 1, pp. 45–68, 2025.
[5] A. Sharma, R. Gupta, and P. Kumar, "AI-Based Resume Screening and Candidate-Job Fit Scoring Using Sentence-BERT and Contextual Embeddings," IEEE Transactions on Computational Social Systems, vol. 11, no. 2, pp. 2104–2115, 2024.

[6] P. K. R. Otani, M. Gan, and S. Warusawithana, "Candidate Profile Summarization and Skill Extraction: A RAG Approach for Recruitment Automation," in Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 112–126, 2024.
[7] D. Lewis, H. Park, and T. Zhang, "Dense Passage Retrieval and Vector Indexing via FAISS for Real-Time Talent Acquisition Systems," IEEE Access, vol. 12, pp. 45890–45902, 2024.
[8] M. H. R. Al-Mulla, S. Chen, and A. Al-Haddad, "LEMON: LLM-Enabled Orchestration and Resilience Monitoring for Distributed Microservices," IEEE Transactions on Services Computing, vol. 17, no. 3, pp. 889–901, 2025.
[9] R. V. Kulkarni, B. S. Patil, and N. V. Deshmukh, "A Modular Multi-Agent RAG System for Intelligent Document Analysis and Intent-Based Dynamic Routing," International Journal of Research in Engineering and Technology, vol. 13, no. 1, pp. 45–58, 2024.
[10] S. A. Miller, K. L. Davis, and J. M. White, "Comparative Evaluation of RAG Architectures: Mitigating Rate Limits and Latency Spikes in Hybrid AI Systems," ACM Transactions on Intelligent Systems and Technology, vol. 16, no. 2, art. 24, pp. 1–22, 2025.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY