IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, July 2026
Impact Factor: 8.2

Predictive Talent Intelligence Platform for Smart Hiring Decisions

B Eshwari[1], Harshitha C S[2], Dr. Kruthi R[3]
[1] Department of Master of Computer Applications
[2] Associate Professor, Department of Master of Computer Applications
Maharaja Institute of Technology, Mysuru, Karnataka, India

Abstract: *Talent acquisition remains one of the most resource-intensive functions in any modern organization. HR departments routinely sift through hundreds of resumes, conduct multiple interview rounds, and still face last-minute offer rejections that derail project timelines and inflate hiring costs. Existing Applicant Tracking Systems (ATS) do little more than store candidate data they offer no predictive capability to help recruiters foresee whether a candidate will actually join after accepting an offer. This paper introduces TalentIQ, a web-based intelligent recruitment automation platform.*

Keywords: *Talent acquisition*

I. INTRODUCTION

The challenge of building the right team has always been central to organizational success, yet the actual process of recruitment remains remarkably inefficient in most companies. HR professionals deal with a constant flood of applications, many of which are poorly structured or difficult to compare objectively. The problem does not end at screening even after rounds of interviews and salary negotiations, a significant number of candidates withdraw or reject the offer at the last stage. This phenomenon, commonly referred to as offer dropout, is particularly damaging in the IT and services industry where hiring timelines are tightly coupled with project kickoffs.

Traditional Applicant Tracking Systems were designed to address administrative overload, and they do that reasonably well. But they stop short of being truly intelligent. They do not analyze what is inside a resume beyond keyword tags. They cannot tell a recruiter which candidate, among a shortlisted group, is statistically more likely to show up on the first day of work. They do not process the qualitative notes that interviewers write about candidates in any meaningful way. The result is that even experienced HR teams are often making critical decisions on instinct rather than evidence.

Artificial Intelligence and Machine Learning have matured enough to address exactly these gaps. Supervised classification models can learn patterns from historical hiring data and generate probabilistic predictions about future behavior. Natural language processing can convert paragraphs of interviewer commentary into quantifiable sentiment scores. Document parsing libraries can transform an uploaded PDF into a structured data record within seconds.

This paper presents TalentIQ, a system that brings all of these technologies together in a single, practically deployable web application. The system automates the candidate intake process, generates ML-based joining probability predictions, analyzes interview feedback sentiment, and ranks candidates using a weighted composite scoring model. The goal is not to replace human judgment in hiring but to augment it — giving recruiters better information so they can make better decisions faster.

II. RELATED WORK AND RESEARCH GAP

The intersection of machine learning and human resource management has attracted steady research attention over the past decade, with contributions spanning resume screening, attrition prediction, employee performance forecasting, and hiring optimization

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, July 2026
Impact Factor: 8.2

III. METHODOLOGY PROPOSED

TalentIQ is organized as a modular, layered system comprising a client-side interface, a server-side processing engine, a machine learning inference component, and a persistent database backend.

A. Technology Stack

Frontend: Bootstrap 5 HTML/CSS framework, Chart.js for interactive visualizations, JavaScript for client-side form validation.

Backend: Python 3.x with Flask microframework for HTTP routing, SQLAlchemy ORM for database abstraction.

ML & NLP: scikit-learn for Logistic Regression, Decision Tree, and Random Forest classifiers; TextBlob for sentiment analysis; pdfplumber and python-docx for document parsing.

Database: MySQL 8.0 with normalized schema supporting role-based access control.

B. Web Application

The Flask application in app.py defines the routing structure for the entire platform. Jinja2 templates render dynamic HTML pages for the candidate directory, candidate detail views, job requirement management, the analytics dashboard, and the login interface. All routes that access sensitive candidate or prediction data are protected by a session validation decorator that verifies user authentication before processing any request. The frontend uses Bootstrap for responsive grid layouts and Chart.js for rendering the interactive visualizations on the dashboard.

C. Joining Prediction and Candidate Scoring

When a candidate's profile is created, edited, or explicitly refreshed, the system invokes the predict_joining function. This function loads the serialized model and label encoders using joblib.load, encodes the candidate's categorical field values using the pre-fitted encoders to guarantee consistency, and constructs a two-dimensional NumPy feature array matching the training schema. The model's predict_proba method returns the probability of class 1 (joining). A configurable decision threshold, defaulting to 0.5, classifies the candidate as "Will Join" or "Won't Join." Both the probability value and the label are stored in the predictions table and displayed prominently on the candidate's detail page.

The multi-criteria ranking engine computes a composite score for each candidate using four components: skill match percentage (comparing parsed skills against active job requirements), years of experience, interview score, and the ML-predicted joining probability. Each component is multiplied by a recruiter-configurable weight, and the weighted components are summed to produce the final score. Recruiters can adjust the relative weights through the dashboard to reflect different organizational priorities — for instance, weighting joining probability more heavily during a tight project ramp-up window.

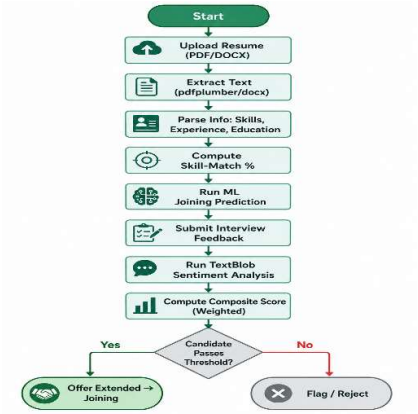
D. Sentiment Analysis, Fraud Flagging, and Dashboard

The sentiment analysis module processes interviewer feedback submissions in real time. When an interviewer submits a comment for a candidate, TextBlob's pattern analyzer evaluates the linguistic polarity of the text and returns a float between -1.0 and +1.0. The system maps this score to one of three labels — Positive, Neutral, or Negative — using fixed thresholds and stores both the score and the label in the interview_feedback table alongside the raw comment text.

The fraud flagging module enables recruiters or administrators to create structured flag records against candidate profiles that exhibit suspicious characteristics — such as implausibly rapid career progression, inconsistencies between the resume and manually entered data, or educational credential anomalies. Each flag carries a severity classification (Low, Medium, High, or Critical) and a descriptive text field for documentation.

The analytics dashboard aggregates data from all tables to present a set of KPI cards showing total candidates, selected count, average joining probability across the pipeline, and model classification accuracy. Below the KPI row, interactive Chart.js charts display monthly hiring trends (applications, selections, and joinings plotted as overlapping lines) and department-wise candidate distribution (rendered as a donut chart). A table at the bottom of the dashboard surfaces at-risk candidates — those with ML-predicted joining probabilities below a threshold — with color-coded probability badges.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, July 2026
Impact Factor: 8.2



IV. SYSTEM ARCHITECTURE

The architecture operates as follows:

Tier 1 — Client Presentation Layer:

The recruiter or HR administrator accesses TalentIQ through any standard web browser. The frontend is built with HTML5, CSS3, and JavaScript. It renders dynamic dashboards, forms, and candidate tables. All user interactions are routed as HTTP requests to the Flask backend.

Tier 2 — Application and Logic Layer (Flask):

The Flask application (app.py) is the heart of TalentIQ. It handles all routing, business logic, session management, and coordination between the different analytical sub-systems. When a resume is uploaded, Flask routes it to the parsing functions. When a candidate profile is viewed, Flask invokes the ML prediction engine. When feedback is saved, Flask calls the sentiment analyzer.

Tier 2a — Analytical Sub-layer:

Resume Parser: Combines pdfplumber/python-docx for text extraction with regex-based pattern matching for structured field extraction.

Sentiment Analyzer: Uses TextBlob to compute polarity scores on interview feedback text.

ML Inference Engine: Loads serialized model files from models/ and executes predict_proba() to generate joining probability estimates.

Tier 3 — Data Persistence Layer:

MySQL serves as the relational database. Six core tables store users, candidates, ML predictions, job requirements, interview feedback, and fraud flags. The Flask application communicates with MySQL using the mysql-connector-python driver.

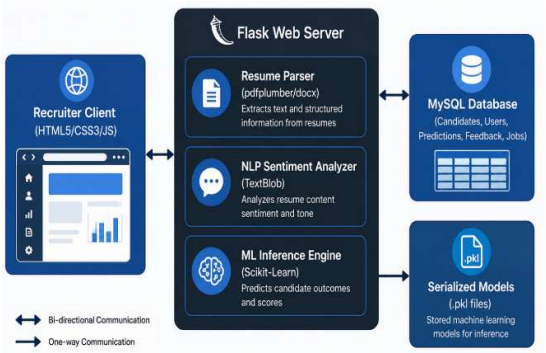


Figure 3.1: System Architecture Diagram of TalentIQ

DOI: 10.48175/IJARSCT-37883

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, July 2026
Impact Factor: 8.2

V. EXPERIMENTAL SETUP

1. Python (Core Programming Language)

Python was selected as the primary programming language for the backend development of TalentIQ due to its extensive ecosystem of scientific computing, machine learning, and natural language processing libraries. Its clean syntax and readability allowed for rapid prototyping of the resume parsing, data preprocessing, and model training pipelines.

2. Flask (Web Framework)

Flask is a lightweight and micro web framework for Python. Unlike heavier frameworks, Flask does not force a particular directory structure or database setup, giving the project the flexibility to integrate custom machine learning pipelines and direct MySQL connection logic. It handles web routing, session authentication, and candidate details administration efficiently.

3. MySQL (Relational Database)

MySQL is used as the relational database management system (RDBMS) to store candidate profiles, system users, recruiter login details, parsed resume text, job requirements, interview feedbacks, and machine learning predictions. It provides structured query execution, relation integrity through foreign keys, and fast retrieval of tabular candidate metrics.

4. Scikit-Learn (Machine Learning Pipeline)

Scikit-Learn is the primary machine learning library used to construct and evaluate the classification models. It is utilized for scaling candidate numerical inputs, encoding categorical features (such as location preference and education level), and running classification models like Logistic Regression and Random Forest to calculate candidate joining probabilities.

5. text Blob (Natural Language Processing)

text Blob is a simple Python library for processing textual data. In this project, it is used to perform sentiment analysis on the qualitative comments submitted by interviewers. By parsing the text and calculating polarity scores, text Blob converts subjective notes into numerical values that are used in candidate ranking.

6. Pdfplumber & Python-Docx (Resume Parsing)

Pdfplumber: This library is used to read and extract text from PDF documents. It is chosen for its accuracy in preserving word layouts and spacing, which is crucial when parsing structured resume sections using regular expressions.

Python-Docx: This package is used to read and process Word documents (.docx), ensuring that candidate profiles submitted in Microsoft Word format are parsed just as effectively as PDFs.

7. Pandas & NumPy (Data Manipulation)

Pandas: Used for data analysis and preparing the candidate training dataset. It simplifies structural operations, data cleaning, and dataset exporting.

NumPy: Handles large-dimensional arrays and matrix computations behind the scenes during feature vector transformations and ML model inference.

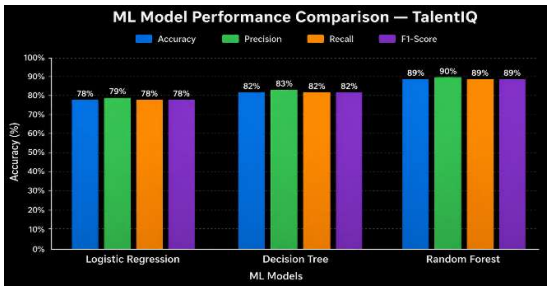
VI. CONCLUSION AND DISCUSSION

TalentIQ was built around one straightforward problem: most recruitment tools can store candidate data but can't do anything useful with it. After completing the build and evaluating the results, it's fair to say the project delivered on what it set out to do. The automated resume parsing engine successfully eliminates what is perhaps the most tedious manual task in early-stage recruitment — reading and extracting structured data from uploaded documents. By combining pdfplumber for PDF files and python-docx for Word documents with a regex-based extraction pipeline, the system handles the diverse formatting reality of real-world resumes with reasonable accuracy.

The machine learning joining prediction module represents the most analytically significant component of the project. By training a Random Forest Classifier on historical candidate data and serializing it for real-time inference, TalentIQ provides recruiters with a probabilistic signal that was previously unavailable — an estimate of whether a given candidate, based on their profile characteristics, is likely to actually show up on the joining date. The model's accuracy of

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, July 2026
Impact Factor: 8.2

approximately 89 percent on the test set demonstrates that this prediction is meaningful and reliable enough to be used as a real decision-support tool.



VII. CONCLUSIONS

TalentIQ addresses a well-defined and practically significant problem: the difficulty of making fast, objective, and accurate hiring decisions in high-volume recruitment environments where candidate data is unstructured, evaluations are subjective, and offer outcomes are uncertain. The system achieves this by combining four complementary technologies — document parsing, machine learning classification, natural language processing, and weighted multi-criteria decision-making — into a single, coherent web application that non-technical HR professionals can use without assistance.

The automated resume parser eliminates the manual effort of extracting and entering candidate information, reducing the time from application receipt to profile creation to a matter of seconds. The machine learning pipeline provides a statistical, evidence-based estimate of joining probability that complements rather than replaces recruiter intuition. The sentiment analysis module surfaces the emotional tone embedded in qualitative interviewer feedback, transforming an otherwise discarded data source into a quantifiable signal. The ranking engine gives recruiters a tool to impose their own organizational logic on the candidate shortlist, without requiring any programming knowledge.

REFERENCES

[1] A. Sharma and R. Mehta, "Applicant Tracking Systems with Machine Learning Integration," Journal of HR Technology, vol. 12, no. 2, pp. 45–53, 2023.

[2] S. Kumar and L. Patel, "Predicting Employee Joining Behavior in the IT Sector," International Conference on Data Science and Engineering, pp. 112–119, 2024.

[3] V. Prasad and J. Rao, "Resume Parsing Using Natural Language Processing Techniques," IEEE Transactions on Systems, Man, and Cybernetics, vol. 8, pp. 201–207, 2022.

[4] M. Al-Mansoori, "Sentiment Analysis on Interview Feedbacks Using TextBlob," International Journal of NLP Applications in Business, vol. 5, no. 1, pp. 88–95, 2025.

[5] H. Dupont, "Multi-Criteria Decision Making in Human Resource Selection," European Journal of Human Resource Management, vol. 19, no. 4, pp. 312–318, 2023.

[6] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.

[7] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[8] R. Bird, E. Klein, and E. Loper, Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit, O'Reilly Media, 2009