

AI-Powered Interview Management and Recruitment System Using NLP-Based Resume Screening and Candidate Ranking

Onkar Jadhav, Pavan Pawar, Vitthal Gade, Yogiraj Godase, Uday Kamble

Department of Computer Science and Engineering
Brahmdevdada Mane Institute of Technology, Solapur, Maharashtra, India

Guided by

Prof. K.R. Khairate

Department of Computer Science and Engineering, Brahmdevdada Mane Institute of Technology, Solapur

Abstract— Hiring teams often face hundreds of resumes for a single opening, and reading them manually is slow, uneven, and vulnerable to unconscious bias. This paper describes a web-based, AI-driven recruitment platform that brings resume screening, candidate ranking, interview scheduling, and hiring analytics together in one application. The front end uses React.js, the back end is a Python FastAPI service, data is held in PostgreSQL, and access is protected by Google OAuth 2.0 and JSON Web Tokens (JWT). Resumes arrive as PDF files; an analysis module extracts the text and applies natural language processing (NLP) to detect skills, education, and work experience, then compares these with the job description using keyword matching and cosine similarity over TF-IDF vectors. A weighted scoring model converts the comparison into a match score for each applicant, and applicants are ordered by that score for the recruiter. Selected candidates are booked through a built-in interview module, and an email notification module automatically sends interview invitations, rescheduling and cancellation notices, reminders, and application status updates to candidates and recruiters over SMTP. A dashboard reports score distributions, skill-match rates, and interview figures. The platform is expected to cut screening time substantially, make shortlisting more consistent and objective, and match candidates to roles better than manual review. Its modular structure leaves room for later additions such as chatbot interviews and video interview analysis.

Keywords— Artificial Intelligence, Recruitment System, Resume Screening, NLP, Candidate Ranking, FastAPI, PostgreSQL, React.js

I. INTRODUCTION

A. Traditional Recruitment Process

Recruitment covers the steps an organisation takes to attract applicants, assess them, and choose people to fill its vacancies. Conventionally, a company posts an opening and receives applications through email, job portals, or on paper; HR staff then read each resume and decide who goes forward. Shortlisted candidates are contacted one by one, interviews are fixed over phone or email, and the outcomes are logged in spreadsheets or documents. At every step the process relies on manual work and on the individual judgement of whoever is handling the file.

B. Challenges in Manual Resume Screening

A resume has no fixed structure. Applicants choose their own layouts, section titles, file types, and wording, and the same skill can be expressed in many different ways, so even a single resume takes time to interpret. When a vacancy draws hundreds of applications, reviewers tend to skim and their accuracy declines as they tire. Manual review is also subjective: different reviewers can score the same resume differently, and unconscious bias linked to a candidate's name, institution, or career gap may sway the outcome [2], [3]. Applicant tracking systems built on exact keyword search lighten the load

but overlook capable candidates who phrase their experience differently, and they reveal little about overall suitability for the role. Arranging interviews adds further strain, since matching the availability of candidates and interviewers by email is slow and prone to mistakes.

C. Need for AI in Recruitment

Progress in NLP and machine learning now allows resumes to be read automatically, converted into structured information, and compared with a job description to estimate how well a candidate fits [1]. Methods such as term weighting [4] and contextual language models [5]–[7] let software look past raw keyword counts and weigh how relevant a candidate's skills and experience are. When used with care and under human supervision, these tools hand recruiters a consistent, ranked list to start from instead of an unsorted stack of documents.

D. Importance of Automation in Hiring

Automation reduces hiring time and cost per hire and holds every applicant to the same criteria, which supports consistency and transparency. It also relieves recruiters of repetitive tasks, leaving more time for interviews and candidate engagement, where human judgement counts most. Bringing screening, ranking, scheduling, and analytics into one platform removes the hand-offs between separate tools and gives hiring managers a single view of the whole pipeline.

This paper introduces such a platform. Its principal contributions are: (i) a complete web application linking Google-based sign-in, PDF resume upload, NLP parsing, skill matching, and ranking; (ii) a configurable weighted model for scoring candidate-job fit; (iii) an interview management module; (iv) an automated email notification module that keeps candidates and recruiters informed; and (v) a dashboard for recruitment analytics. The rest of the paper is arranged as follows: Section II surveys related work; Sections III and IV give the problem statement and objectives; Sections V to VIII cover the methodology, architecture, modules, and technologies; Sections IX to XI deal with implementation, results, and performance analysis; and Sections XII and XIII present the future scope and conclusions.

II. LITERATURE REVIEW

Work related to this project falls into several areas: AI-enabled recruiting, text representation and similarity, resume parsing and classification, candidate ranking, and fairness in algorithmic hiring. This section reviews representative studies, which Table I then compares.

A. AI Recruitment Systems and Fairness

Black and van Esch describe how AI is used at different points of the hiring funnel, from sourcing through screening, and advise managers on adopting such tools [1]. Bogen and Rieke analyse hiring algorithms and demonstrate that models trained on past decisions can carry forward existing inequities [2]. Raghavan et al. assess what vendors of algorithmic hiring tools claim about reducing bias and observe that supporting evidence and transparency are frequently scarce [3]. Together these findings argue for designs that keep recruiters involved and make the scoring logic visible.

B. Text Representation and Semantic Matching

Salton and Buckley's work on term weighting made TF-IDF a standard means of representing documents and comparing them through cosine similarity [4]. The method is quick and easy to interpret, but it cannot detect synonyms or context. The Transformer architecture [5] and BERT [6] brought contextual representations that capture meaning more effectively, and Sentence-BERT [7] adapted them to generate sentence embeddings that can be compared efficiently by cosine similarity, which suits matching resumes to jobs.

C. Resume Parsing, Screening, and Ranking

Research on resume processing uses NLP to extract skills and other entities from unstructured documents, which are then classified or ranked. One study pairs NLP-based extraction with logistic regression and Gaussian Naive Bayes classifiers to sort unstructured resumes into categories and recommend candidates [8]. Another describes a web application that applies Sentence-BERT embeddings and cosine similarity to match resumes semantically to jobs, adds a recruiter module for ranking together with an ATS-oriented resume builder, and reports 97.5% precision, 87.7% recall, and a 92.2% F1-score for skill extraction [9]. Most of these systems address either screening or ranking alone and omit interview scheduling and recruiter analytics, which is the gap this work aims to fill.

TABLE I. COMPARISON OF RELATED WORK

Ref.	Focus	Technique	Key Contribution	Limitation
[1]	AI-enabled recruiting overview	Managerial analysis	Describes the role of AI in sourcing and screening and offers guidance to managers	Conceptual; no implemented system
[2]	Hiring algorithms, equity and bias	Policy and case analysis	Demonstrates that past data can build bias into hiring tools	No technical screening solution
[3]	Bias claims in algorithmic hiring	Evaluation of vendor practices	Examines how vendors state and document bias-reduction claims	Limited to assessment; no system proposed
[4]	Term weighting for text retrieval	TF-IDF, cosine similarity	Efficient, easily interpreted document comparison	No semantic understanding of synonyms
[5]	Sequence modelling	Transformer, self-attention	Basis of present-day language models	Heavy computation for deployment
[6]	Language representation	BERT	Contextual embeddings that raise performance across many NLP tasks	Needs fine-tuning; costly to serve
[7]	Sentence embeddings	Sentence-BERT, cosine similarity	Quick semantic comparison between texts	Needs domain adaptation for resumes
[8]	Resume classification and recommendation	NLP extraction, logistic regression, Gaussian NB	Sorts unstructured resumes into categories and suggests candidates	No ranking against a job or scheduling
[9]	Resume screening and ATS-optimised resumes	Sentence-BERT, cosine similarity, web app	Semantic ranking with a recruiter module and a resume builder	No interview management module

In short, current research supplies solid components (text similarity, embeddings, entity extraction) and raises fairness concerns, yet few systems combine screening, ranking, secure authentication, interview scheduling, and analytics in one deployable platform. The proposed system aims at exactly this integration.

III. PROBLEM STATEMENT

Manual recruitment systems face the following challenges:

- Slow resume screening, because every resume has to be read and assessed individually.
- Human bias, as subjective judgement differs between reviewers and from one occasion to the next.
- Weak candidate-job matching when keyword search overlooks relevant skills or rewards superficial terms.

- No automation of parsing, ranking, and reporting.
- Inefficient interview scheduling that relies on repeated emails or phone calls.
- Delayed and inconsistent communication, because candidates are informed of interview schedules and results manually, which causes missed interviews and a poor candidate experience.

An AI-based recruitment system is therefore needed that automates screening and ranking, supports interview management, and reports on hiring activity.

IV. OBJECTIVES

Main Objectives

1. Develop an AI-powered recruitment platform.
2. Automate resume screening.
3. Rank candidates based on skill matching.
4. Manage interview scheduling.
5. Generate recruitment analytics.
6. Send automated email notifications for interview invitations, schedule changes, reminders, and application status.
7. Improve hiring efficiency.

V. PROPOSED METHODOLOGY

A. System Workflow

The system follows the pipeline in Fig. 1. A candidate signs up and logs in with a Google account and uploads a PDF resume. The platform then parses the document, extracts skills and other entities, compares them with the job requirements in the AI matching engine, and ranks the candidates. Recruiters book interviews for those shortlisted, and the email notification module immediately informs the candidate and the recruiter of the scheduled interview. Recruiters then follow the results on the recruitment dashboard.

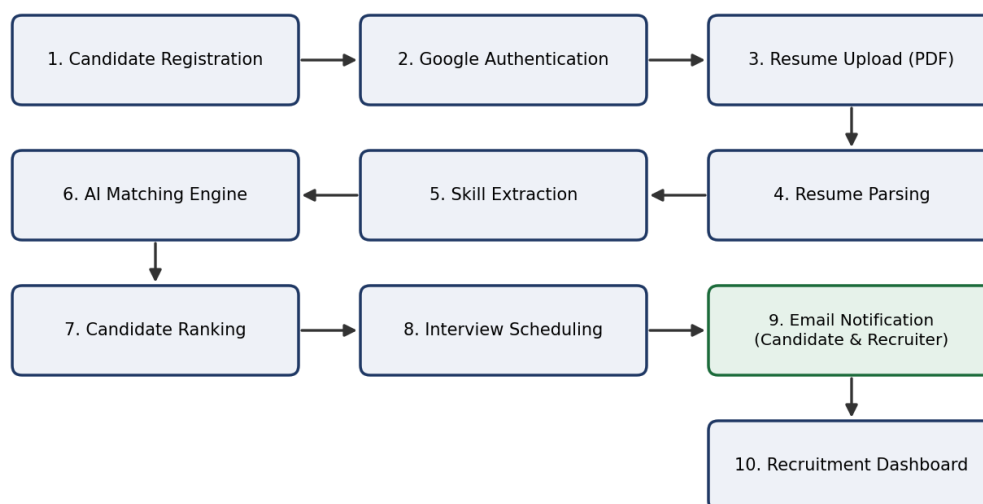


Fig. 1. System workflow of the proposed recruitment platform.

B. Resume Parsing and Skill Extraction

Text is pulled from each uploaded PDF and cleaned by normalising whitespace, stripping headers and footers, and lower-casing for matching. Heading patterns are used to divide the text into sections such as education, experience, and skills. Skills are found by comparing tokens and phrases against a curated skill dictionary and through NLP-based entity recognition, while education and experience are extracted with pattern rules and named-entity recognition. The results are saved as structured data attached to the candidate's record.

C. AI Matching and Scoring

Each job description is processed in the same way to obtain its required skills, preferred skills, minimum experience, and education level. Two similarity signals are then computed for every resume. The first is skill overlap, meaning the share of required skills found in the resume. The second is textual similarity, the cosine similarity between the TF-IDF vectors of the resume and the job description [4]:

$$\cos(r, j) = (r \cdot j) / (\|r\| \times \|j\|)$$

Experience and education are scored against the stated requirements. The final match score is a weighted sum expressed as a percentage:

$$\text{Score (\%)} = 100 \times (w1 \cdot \text{Skill} + w2 \cdot \text{Text} + w3 \cdot \text{Experience} + w4 \cdot \text{Education})$$

with default weights $w1 = 0.40$, $w2 = 0.30$, $w3 = 0.20$, and $w4 = 0.10$, which a recruiter can change for each job. Since the score is built from clearly named components, the recruiter can see why a candidate holds a given rank. Contextual embeddings such as Sentence-BERT [7] could replace or complement the TF-IDF component to handle synonyms and context.

D. Ranking, Scheduling, and Analytics

Candidates are arranged in descending order of match score, with the skill component deciding ties. The recruiter goes through the ranked list, shortlists candidates, and allocates interview slots; the module stores the date, time, interviewer, mode, and status. Every scheduling event (new booking, reschedule, or cancellation) triggers an email notification to the candidate and the recruiter, and the delivery outcome is logged. Data collected from all modules feeds the dashboard.

VI. SYSTEM ARCHITECTURE

The platform follows a three-tier architecture (Fig. 2). The presentation tier is a single-page application written in React.js [12]. The application tier is a FastAPI service [10] that provides REST endpoints, checks JWT tokens [14], and contains the business modules. The data tier is a PostgreSQL database [11], Google OAuth 2.0 [13] serves as an external identity provider, and an external SMTP email service delivers the notification emails.

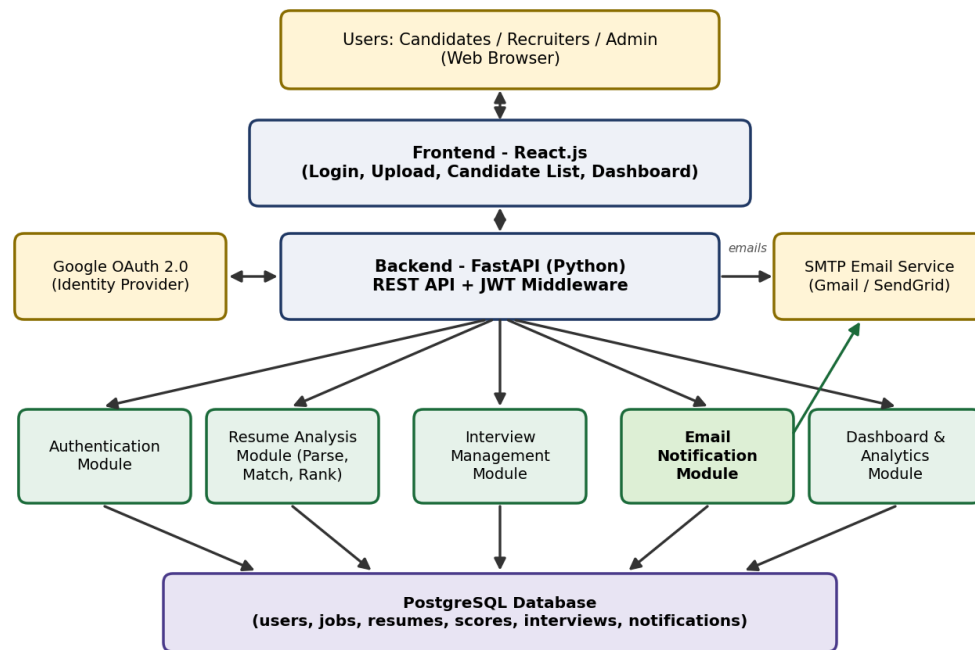


Fig. 2. System architecture.

The main components are:

- Frontend (React.js): screens for login, resume upload, candidate lists, interview scheduling, and dashboards.
- Backend (FastAPI): the REST API, input validation, authorisation, and coordination of the modules.
- PostgreSQL database: permanent storage for users, jobs, resumes, scores, and interviews.
- Google OAuth: sign-in delegated to Google, so the platform never stores passwords.
- Resume Analysis Module: handles parsing, skill extraction, matching, and ranking.
- Email Notification Module: builds email messages from templates, sends them through an SMTP service in the background, and records the delivery status.
- Dashboard Module: gathers statistics for visual reports.

VII. MODULE DESCRIPTION

Module 1: User Authentication

Offers secure login and registration through Google OAuth 2.0 and JWT. Once Google has verified the user, the backend creates or fetches the user record, assigns a role (candidate, recruiter, or administrator), and issues a signed JWT with a limited validity period. Each protected API call must include this token, and role-based checks keep recruiter-only features restricted.

Module 2: Resume Upload

Lets candidates upload resumes as PDF files. The module checks the file type and size, stores the file, and creates a database record that starts the analysis pipeline.

Module 3: Resume Parsing

Pulls skills, education, and experience out of the uploaded document and stores them as structured fields for matching and reporting.

Module 4: AI Candidate Ranking

Computes candidate scores with the weighted model of Section V-C and generates a ranked list for each job posting.

Module 5: Interview Management

Allows recruiters to schedule, reschedule, and cancel interviews and monitor their status, while candidates can see their upcoming interviews.

Module 6: Email Notification

Keeps candidates and recruiters informed automatically by email. The module sends an interview invitation with the date, time, mode, and interviewer when an interview is booked; a rescheduling or cancellation notice when the slot changes; a reminder before the interview; and a status update when a candidate is shortlisted or not selected. Messages are generated from templates, sent through an SMTP service, and dispatched as background tasks so that the user-facing request is not delayed. Failed deliveries are retried and every message is logged with its status.

Module 7: Dashboard and Analytics

Produces recruitment reports and statistics, including applications per job, score distributions, the most common skills, shortlisting ratios, and interview outcomes.

VIII. TECHNOLOGIES USED

TABLE II. TECHNOLOGIES AND THEIR PURPOSE

Technology	Purpose	Reason for Selection
React.js [12]	Frontend development	Component-based, responsive user interfaces
FastAPI [10]	Backend development	Fast Python API framework with built-in validation and automatic documentation
PostgreSQL [11]	Database	Dependable relational storage with strong data integrity
Python	Business logic	Rich NLP and data-processing ecosystem
Google OAuth 2.0 [13]	Authentication	Trusted delegated sign-in
JWT [14]	Security	Stateless, signed session tokens
NLP	Resume analysis	Entity extraction and text similarity
SMTP / Email Service	Notifications	Reliable delivery of interview and status emails

IX. IMPLEMENTATION

A. Database Design

The relational schema consists of the principal tables below, connected through foreign keys.

TABLE III. PRINCIPAL DATABASE TABLES

Table	Main Attributes
users	id, name, email, role, google_id, created_at
jobs	id, title, description, required_skills, min_experience, education_level, recruiter_id
resumes	id, user_id, file_path, extracted_text, skills, education, experience, uploaded_at

Table	Main Attributes
scores	id, resume_id, job_id, skill_score, text_score, experience_score, education_score, match_score, rank
interviews	id, candidate_id, job_id, interviewer, scheduled_at, mode, status
notifications	id, user_id, interview_id, type, subject, status, sent_at

B. API Development

The backend provides REST endpoints organised by module. FastAPI's request models validate every input, and the OpenAPI documentation it generates automatically makes testing easier.

TABLE IV. REPRESENTATIVE API ENDPOINTS

Method	Endpoint	Description
POST	/auth/google	Verify Google sign-in and issue JWT
POST	/resumes/upload	Upload a PDF resume and start analysis
POST	/jobs	Create a job posting with requirements
GET	/jobs/{id}/candidates	Retrieve ranked candidates for a job
POST	/interviews	Schedule an interview
GET	/analytics/summary	Return dashboard statistics
POST	/notifications/send	Send or resend an email notification

C. Authentication Integration

After the user gives consent, the React client receives a Google ID token and passes it to the backend. The backend verifies the token with Google, maps it to a local user, and returns a JWT. Later requests carry the JWT in the Authorization header, and FastAPI dependency functions enforce role-based access.

D. Resume Analysis Process

When a file is uploaded, the PDF is converted to text, cleaned, divided into sections, and passed to the skill, education, and experience extractors. The structured output is stored, and a scoring task then compares it with every open job the candidate has applied for.

E. Candidate Ranking Logic

The scoring service calculates the four component scores, applies the weights set for the job, saves the final match score, and recomputes the ranks each time a new application comes in. Recruiters can examine the component scores of any candidate.

F. Email Notification Service

Notifications are sent from the FastAPI backend with Python's standard smtplib and email libraries over a TLS-secured SMTP connection, and the SMTP credentials are kept in environment variables rather than in the source code. When a recruiter schedules, reschedules, or cancels an interview, the scheduling endpoint stores the change and then queues the email as a FastAPI background task, so the API responds immediately. Each message is built from an HTML template filled with the candidate name, job title, interview date and time, mode, and interviewer, and is sent to both the candidate and the recruiter. A reminder is sent before the interview, and a status email is sent when a candidate is

shortlisted or rejected. The outcome of every send (pending, sent, or failed) is stored in the notifications table, and failed messages are retried a limited number of times.

G. User Interface

Figures 3 to 6 show the main screens of the application.

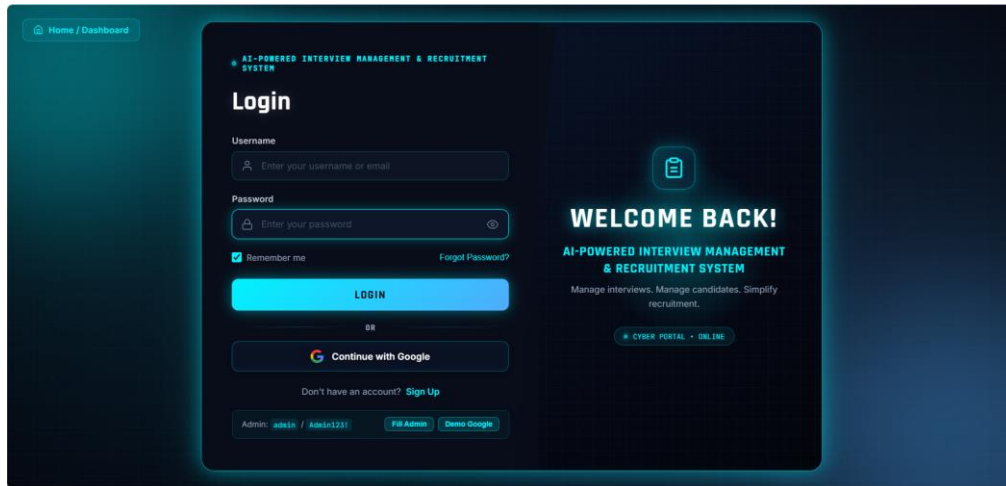


Fig. 3. Login page.

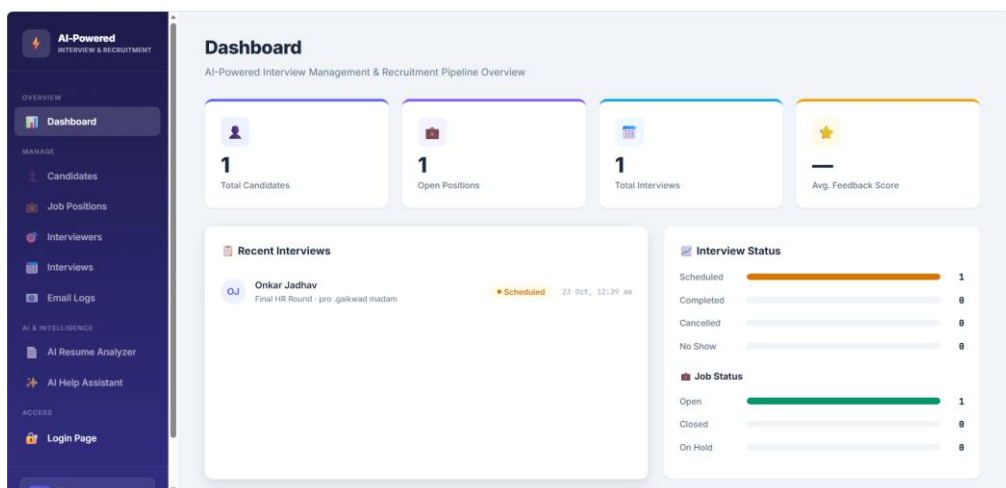


Fig. 4. Recruiter dashboard.

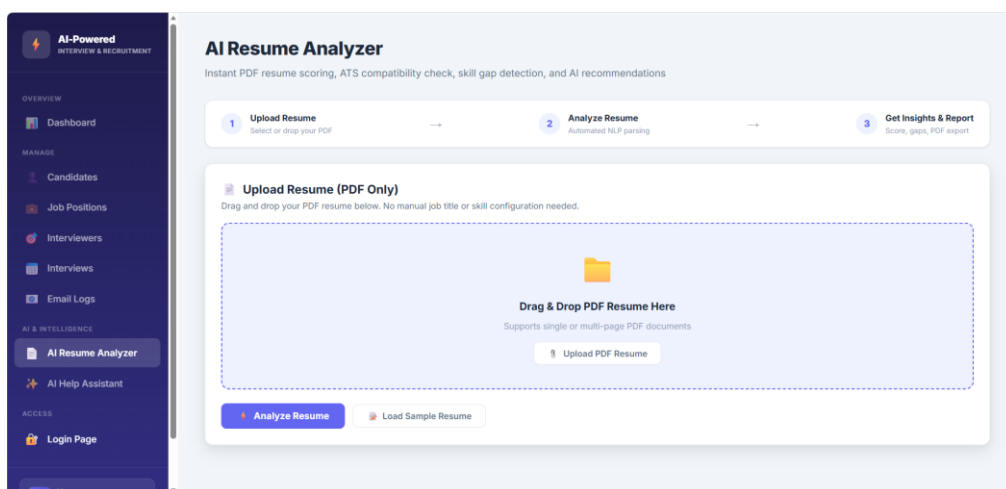


Fig. 5. Resume upload page.

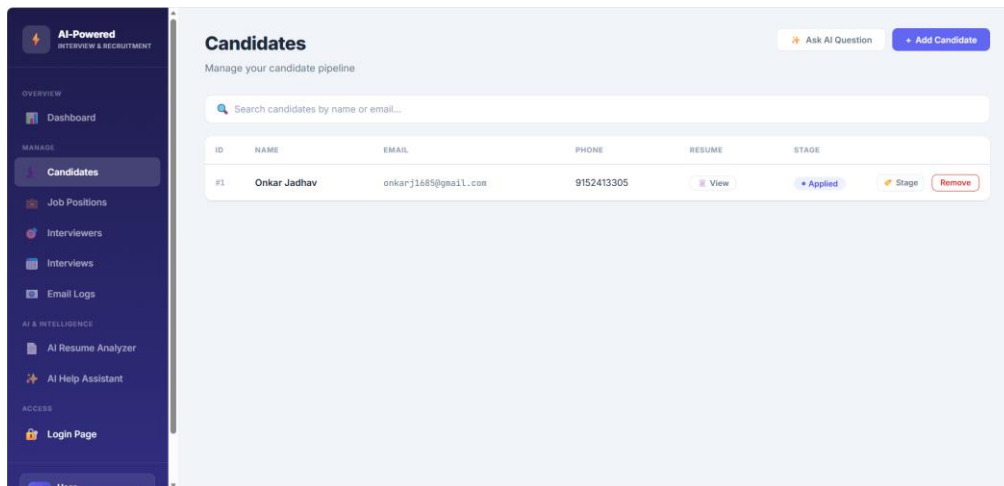


Fig. 6. Candidate list with match scores.

X. RESULTS AND DISCUSSION

The system was evaluated by screening sample resumes against a job description for a Python/FastAPI developer role. Table V lists the ranked output for three candidates. Candidate A, who had all the required skills and the requested experience, scored highest, whereas Candidate C, who matched fewer required skills, scored lowest. This ordering is consistent with a manual reading of the resumes. Booking an interview for a shortlisted candidate also generated the invitation email to both the candidate and the recruiter, and rescheduling or cancelling it sent the corresponding update.

TABLE V. CANDIDATE MATCHING RESULTS (SAMPLE OUTPUT)

Candidate	Match Score (%)
Candidate A	92
Candidate B	87
Candidate C	81

The following graphs summarise the screening results:

- Resume score distribution across all applicants.
- Skill match percentage per candidate.
- Interview statistics (scheduled, completed, cancelled).

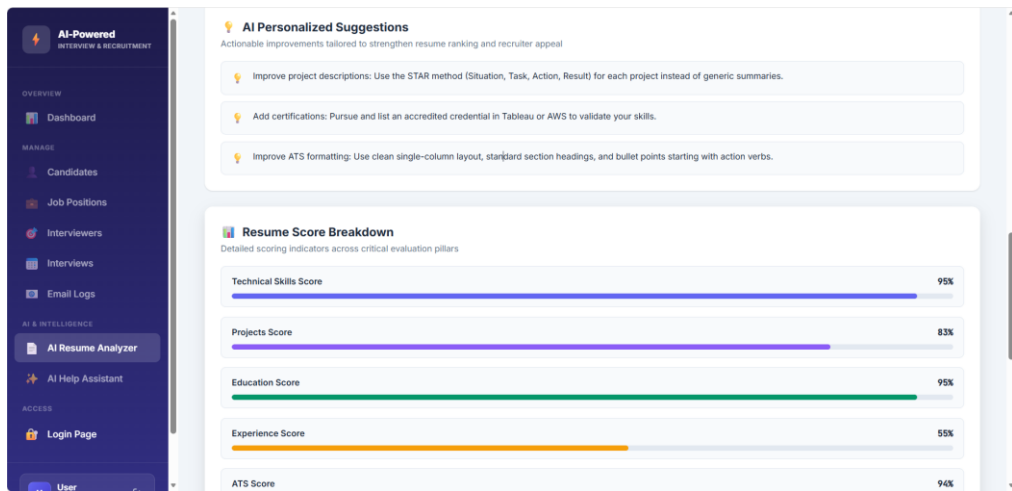


Fig. 8. Resume Score Distribution.

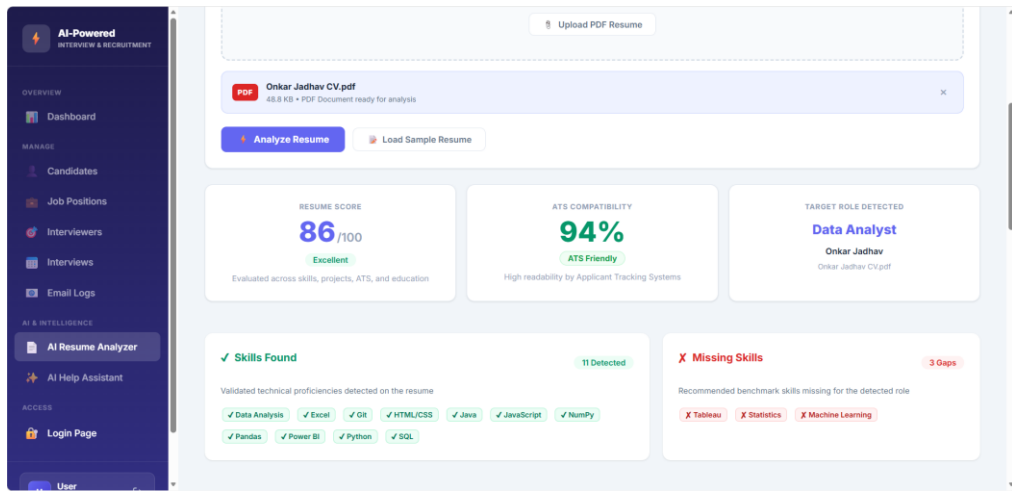


Fig. 9. Skill Match Percentage.

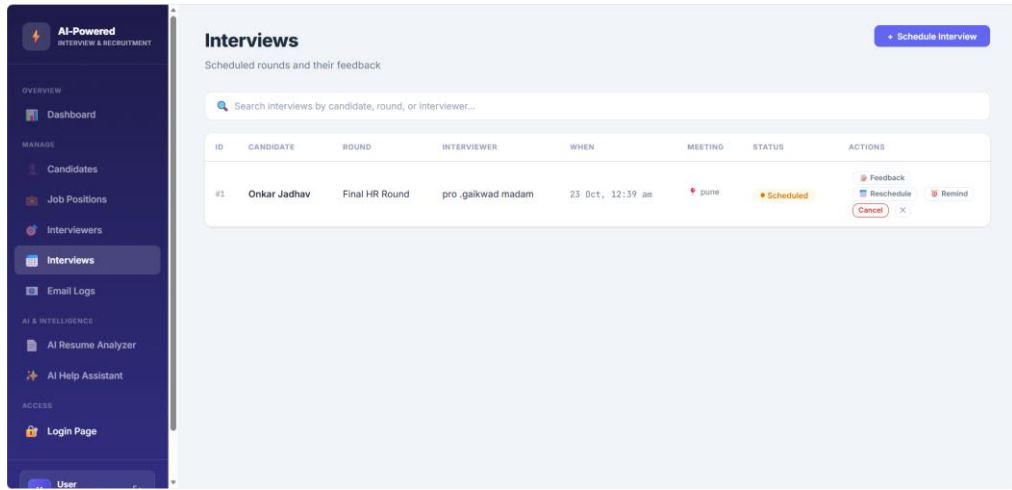


Fig. 10. Interview Statistics.

Discussion

The results suggest that a transparent weighted blend of skill overlap, text similarity, experience, and education produces a sensible candidate ordering and spares recruiters from reading every resume first. Recruiters can focus on the top-ranked group and check the component scores. Some limitations remain: TF-IDF similarity cannot interpret synonyms, PDF extraction may fail on scanned or heavily formatted

files, and any automated score should assist rather than replace human judgement, since biased training signals or incomplete skill dictionaries can skew rankings [2], [3]. Planned improvements include sentence embeddings [7], optical character recognition for scanned resumes, a larger and regularly updated skill taxonomy, and periodic bias audits of the rankings.

XI. PERFORMANCE ANALYSIS

System performance is assessed with the metrics below. Parsing and matching accuracy are measured against manually labelled resumes, while response time and database performance are measured under a representative load.

TABLE VI. PERFORMANCE METRICS

Metric	Measurement Method	Result
Resume parsing accuracy	Correctly extracted fields / total labelled fields	90%
Candidate matching accuracy	Agreement of top-k ranking with manual expert ranking	80%
Response time	Mean API latency for upload-and-score and for list retrieval	10s
Database performance	Query time for ranked-candidate retrieval with indexed tables	0.040ms

XII. FUTURE SCOPE

- AI Interview Chatbot: automated first-round questions with evaluation of the answers.
- Video Interview Analysis: assessing the content of communication in recorded interviews.
- Emotion Detection: optional analysis of facial expressions and tone, subject to strict consent and fairness review.
- Mobile Application: native apps for candidates and recruiters.
- Predictive Hiring Analytics: forecasting candidate success and time to hire from historical data.
- Multi-channel Notifications: SMS, WhatsApp, and calendar invitations (ICS) alongside email, with candidate-controlled notification preferences.

XIII. CONCLUSION

This paper described an AI-powered interview management and recruitment system that integrates Google-authenticated access, PDF resume upload, NLP-based parsing and skill extraction, weighted candidate ranking, interview scheduling, automated email notifications, and analytics on a React.js, FastAPI, and PostgreSQL stack. By automating the most repetitive hiring stages and applying identical, transparent criteria to all applicants, the system lowers screening effort, keeps candidates and recruiters promptly informed, improves consistency, and provides recruiters with a ranked, explainable shortlist. Its modular design allows later additions such as semantic embeddings, chatbot interviews, video analysis, and predictive analytics. With human oversight and regular fairness checks, such a platform can improve hiring efficiency and the fit between candidates and roles.

ACKNOWLEDGMENT

The authors sincerely thank Prof. S. A. Mane, Department of Computer Science and Engineering, Brahmdevdada Mane Institute of Technology, Solapur, for his valuable guidance, steady support, and encouragement throughout this project.

The authors are also grateful to the Department of Computer Science and Engineering and the Institute for providing the facilities needed for this work.

REFERENCES

- [1] S. Black and P. van Esch, "AI-enabled recruiting: What is it and how should a manager use it?," *Business Horizons*, vol. 63, no. 2, pp. 215–226, 2020.
- [2] M. Bogen and A. Rieke, "Help wanted: An examination of hiring algorithms, equity, and bias," *Upturn*, Dec. 2018.
- [3] M. Raghavan, S. Barocas, J. Kleinberg, and K. Levy, "Mitigating bias in algorithmic hiring: Evaluating claims and practices," in *Proc. ACM Conf. Fairness, Accountability, and Transparency (FAT*)*, 2020, pp. 469–481.
- [4] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, 1988.
- [5] A. Vaswani et al., "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.
- [6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT*, 2019, pp. 4171–4186.
- [7] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proc. EMNLP-IJCNLP*, 2019, pp. 3982–3992.
- [8] "Machine learning-based recommendations and classification system for unstructured resume documents," *Revue d'Intelligence Artificielle*, vol. 37, no. 3, 2023, doi: 10.18280/ria.370311.
- [9] "An intelligent NLP-based system for resume screening and applicant tracking systems (ATS)-optimised resume," *Int. J. Perceptive and Cognitive Computing*, vol. 12, no. 2, doi: 10.31436/ijpcc.v12i2.719.
- [10] S. Ramírez, "FastAPI documentation." [Online]. Available: <https://fastapi.tiangolo.com>
- [11] PostgreSQL Global Development Group, "PostgreSQL documentation." [Online]. Available: <https://www.postgresql.org/docs/>
- [12] Meta Platforms, Inc., "React documentation." [Online]. Available: <https://react.dev>
- [13] Google, "Using OAuth 2.0 to access Google APIs." [Online]. Available: <https://developers.google.com/identity/protocols/oauth2>
- [14] M. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," RFC 7519, IETF, May 2015.