iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** X **Month of publication:** October 2026

DOI: https://doi.org/10.22214/ijraset.2026.85463

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 X Oct 2026- Available at www.ijraset.com

InternHub: A Resume-Driven Internship & Job Recommendation Framework with Integrated Scam Screening, Application Tracking and Carrer & Skill Gap Analytics

Mr, Aaksath Nithil Saam SS[1], Mr. Jayabalajivel R[2], Mrs. Yoheswari S[3]
[1, 2]*Student,* [3]*Assistant Professor, Department of Computer Science and Engineering K.L.N. College of Engineering, Pottapalayam, Sivaganga, Tamil Nadu, India*

Abstract: ***Students seeking internships face bloated repositories of postings, frequently plagued by cross-platform duplicates, expired vacancies, and fraudulent stipend schemes. Contemporary literature on internship recommendation primarily concentrates on ranking algorithms under the assumption that pre-structured candidate profiles already exist and that all listings are verified. This paper presents InternHub, an end-to-end web framework that addresses the entire internship discovery and management pipeline. A student uploads an unstructured PDF resume; a dual-condition negative heuristic filter intercepts non-resume documents (e.g., college tuition fee receipts and financial bills); and a natural language processing pipeline extracts a canonical taxonomy of technical skills, project experiences, and target role classifications. In parallel, internship postings are retrieved concurrently across five live job APIs (JobDataAPI, Jooble, Adzuna, Remotive Open API, and Arbeitnow), transformed to a unified schema, pruned of dead links via mirror synchronization, and de-duplicated using composite tuple keys. A rule-based Trust Score Engine screens postings for scam indicators, penalizing upfront fees, unverified chat channels, and truncated specifications. Surviving opportunities are scored and ranked via a mathematically bounded five-factor model (core skills 50%, role alignment 20%, preferred skills 15%, project implementation evidence 10%, work-mode/location preferences 5%), retaining TF-IDF cosine similarity as a lexical baseline. For each opportunity, InternHub conducts a skill-gap analysis, suggests targeted resume refinements, and generates a progressive four-phase learning roadmap. Application states are followed across a six-stage lifecycle (Saved, Applied, Interview, Selected, Rejected, Withdrawn). The platform is implemented using React 19, TypeScript, Express.js, a FastAPI NLP microservice (PyMuPDF, spaCy, scikit-learn), and Supabase PostgreSQL with Row-Level Security (RLS). Functional verification across an 18-scenario automated suite achieved a 100% pass rate, with full end-to-end recommendation requests averaging 1.84 s.***

Keywords: ***Internship recommendation system, resume parsing, natural language processing, content-based filtering, TF-IDF cosine similarity, fake job detection, skill-gap analysis, application lifecycle tracking.***

I. INTRODUCTION

For engineering undergraduates, internships represent the vital transition from theoretical academic curricula to professional software engineering practice. However, finding suitable openings has devolved into an arduous filtering ordeal. Job portals, company career sites, and aggregators release thousands of listings daily. A candidate specialized in Python and machine learning must sift through unrelated postings for sales, accounting, and business development before finding applicable roles. Recommender systems offer an established approach to combat information overload [1], [2], with several recent works investigating internship recommendation [3]–[7]. Nevertheless, existing literature overwhelmingly focuses on the ranking phase alone: given an existing structured candidate profile and a clean catalog of listings, rank the postings. This assumes two critical conditions that rarely hold in real-world environments:

1) That candidates already possess structured digital profiles (whereas students actually maintain PDF resumes and find manual form entry tedious and error-prone).
2) That every opportunity listed on public portals is authentic and safe (ignoring the well-documented surge in recruitment fraud, upfront registration charges, and stipend scams [8], [9]).

©IJRASET: All Rights are Reserved

A student relying on traditional rankers must still manually re-enter profile data, verify listing authenticity, determine missing competencies, and independently monitor submitted applications.

InternHub addresses this comprehensive sequence through five core contributions:

- Resume-First Profile Ingestion: Automated candidate profiling from unstructured PDF documents with an upfront negative heuristic filter that discards irrelevant uploads (such as college fee receipts or invoices).
- Integrated Scam Screening & Trust Scoring: Proactive inspection and de-duplication of live postings from five external job providers, computing an explainable Trust Score (15–98) that flags deceptive traps before recommendation ranking.
- Bounded, Multi-Factor Explainable Matching: A five-factor formulation bounded in [0, 98] that breaks down recommendations into core competencies, title semantics, preferred qualifications, verified project evidence, and candidate preferences, complemented by TF-IDF cosine similarity as a baseline.
- Post-Ranking Skill Gap & Dynamic Roadmap: Automated extraction of missing competencies against target roles, generating a four-phase learning curriculum to guide candidate upskilling.
- Decoupled Architecture with Strong Isolation: Microservice separation across modern UI layers, API gateways, Python NLP processors, and a Supabase PostgreSQL datastore enforced by database-level Row-Level Security (RLS).

II. RELATED WORK

A. Recommendation Techniques

Content-based filtering recommends items whose descriptions align with user profiles [10], [11]. Raw text is mapped to weighted term vectors using TF-IDF [12], [13], with similarity evaluated via cosine angle formulations [14]. Because content-based models depend solely on user attributes, they effectively mitigate the cold-start problem inherent in collaborative filtering approaches that require historical user-item interactions [15], [16]. Hybrid systems integrate both paradigms to balance predictive power [17], but for nascent platforms with zero interaction telemetry, content-based mechanisms remain the most viable starting point.

B. Internship Recommendation Systems

Recent studies explore tailored internship matching. Geethanjali et al. [3] implemented a MERN-stack engine for the PM Internship Scheme utilizing JavaScript-based skill intersection paired with LLM-generated justifications, reporting a Precision@5 of 0.84 on a closed test set of 50 students and 60 jobs. TechTrack [4] leveraged machine learning on student academic records and coursework to suggest career pathways. Mol and Manoj [5] developed normalized feature vector matching, observing potential demographic bias when training data are unbalanced. Victoria et al. [6] engineered a national scheme matcher addressing candidate-to-institutional profile mapping. Busad et al. [7] designed a linear scoring architecture (FinalScore = $\alpha \cdot$SkillScore + $\beta \cdot$AcademicScore) reliant on manually typed comma-separated skill inputs.

A common limitation across these systems is their dependency on static, manually entered inputs and pre-curated institutional datasets rather than live, real-time job feeds and unstructured resume documents.

C. Online Recruitment Fraud

Recruitment fraud has primarily been studied as an isolated text classification task. Vidros et al. [8] developed the Employment Scam Aegean Dataset (EMSCAD) to benchmark fraud classifiers, while Alghamdi and Alharby [9] evaluated multiple supervised models on recruitment corpora. However, these scam detection mechanisms have not previously been synthesized directly inside an end-to-end student recommendation pipeline.

D. Comparative Positioning

Table I highlights InternHub's functional coverage relative to commercial portals and surveyed academic literature.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

TABLE I. FUNCTIONAL COVERAGE COMPARISON

Functional Capability	Commercial Portals	Surveyed Literature [3]–[7]	InternHub (Proposed)
Profile Generation	Manual form entry	Manual entry / CSV dataset	Automated PDF Parsing (PyMuPDF, spaCy)
Non-Resume Bill Rejection	No	Not described	Yes (19-Pattern Heuristic Guardrail)
Multi-Provider Live Feeds	Proprietary listings	Static closed database	Yes (5 Concurrent APIs)
De-duplication & Dead Purge	Rare / Ad-hoc	Not described	Yes (Tuple Hashing & Mirror Sync)
Scam Detection / Trust Score	Reactive reporting	Not described	Yes (Proactive Rule Engine, 15–98 Score)
Multi-Factor Ranking	Keyword / Boolean	Weighted linear / LLM	Yes (5-Factor Explainable Bounded Score)
Skill-Gap & Learning Roadmap	No	Limited	Yes (Set Difference & 4-Phase Roadmap)
Application Tracker	External tool required	Not described	Yes (6-State In-Platform Tracker)

III. SYSTEM ARCHITECTURE

InternHub is built from decoupled services (Fig. 1). A React 19 single-page application (TypeScript, Vite, Tailwind CSS v4, Recharts) talks over HTTPS, with a JWT bearer token, to a Node.js and Express gateway. The gateway checks the session, validates input, queries the five job APIs at the same time, removes duplicates, computes the Trust Score and ranks the listings. Language processing runs in a separate Python 3.11 FastAPI service that handles PyMuPDF text extraction, the non-resume filter, skill normalisation, role inference over more than 35 software roles, and scikit-learn TF-IDF. Persistent data and authentication are provided by Supabase PostgreSQL.

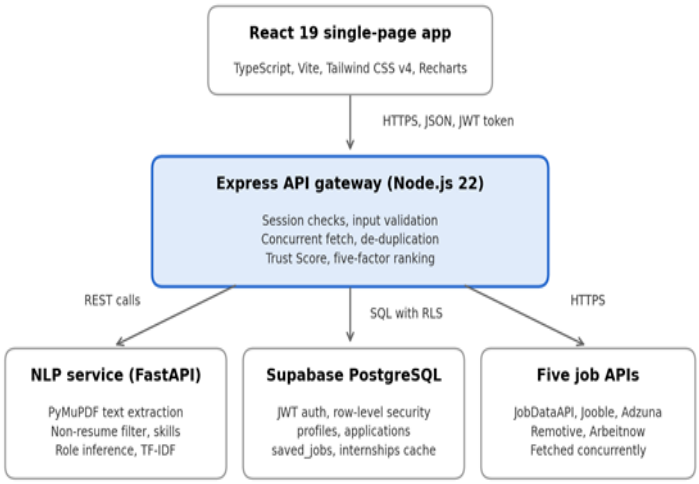


Fig. 1. Decoupled architecture of InternHub. The Express gateway coordinates the NLP service, the database and the five job APIs.

A. Data Isolation and Security

Four tables hold user data: profiles, applications, saved_jobs and a cached copy of aggregated internships. Student records are isolated with PostgreSQL row-level security (RLS) policies tied to auth.uid(), so the database itself refuses a request for another student's rows. A trigger, handle_new_user(), creates a profile row as soon as an identity is created. Uploaded documents are processed in memory and are not kept as raw files without the student's consent.

B. Implemented Application Interface

Figs. 2–8 show the implemented InternHub web interface, in the order of the user workflow.

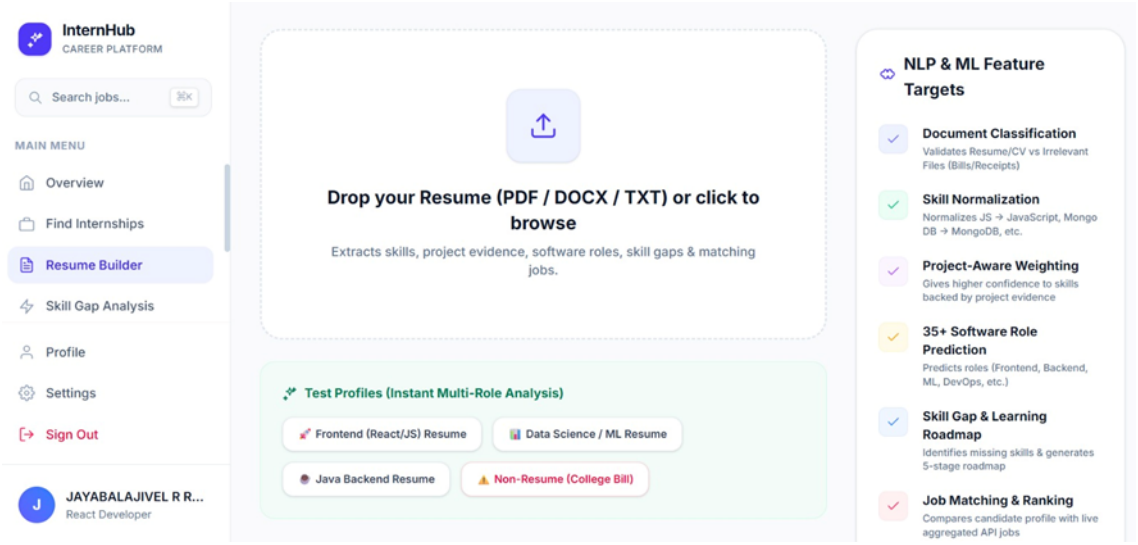


Fig. 2. InternHub resume upload interface.

The upload area accepts PDF, DOCX or TXT resumes and offers test profiles, including a non-resume college bill; a side panel lists the NLP and ML feature targets.

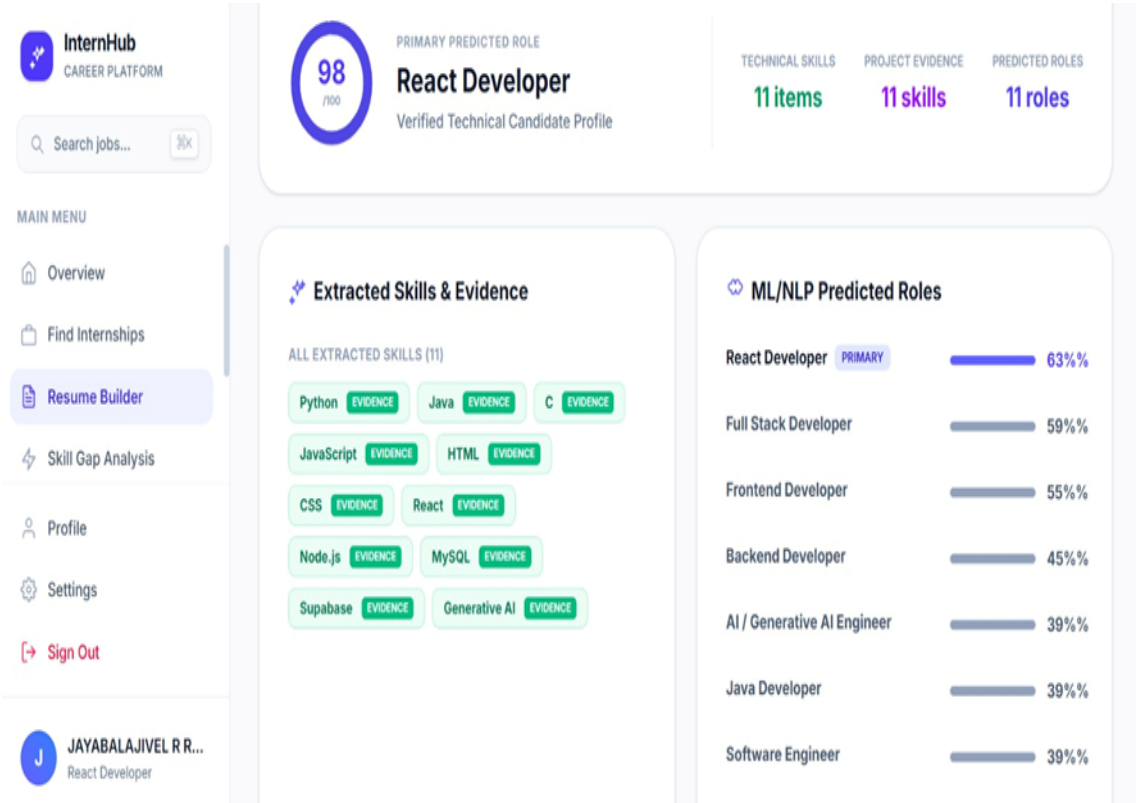


Fig. 3. Resume analysis with extracted skills and predicted software roles.

The page shows a 98/100 score with React Developer as the primary predicted role, 11 extracted skills marked with evidence, and ranked ML/NLP role predictions.

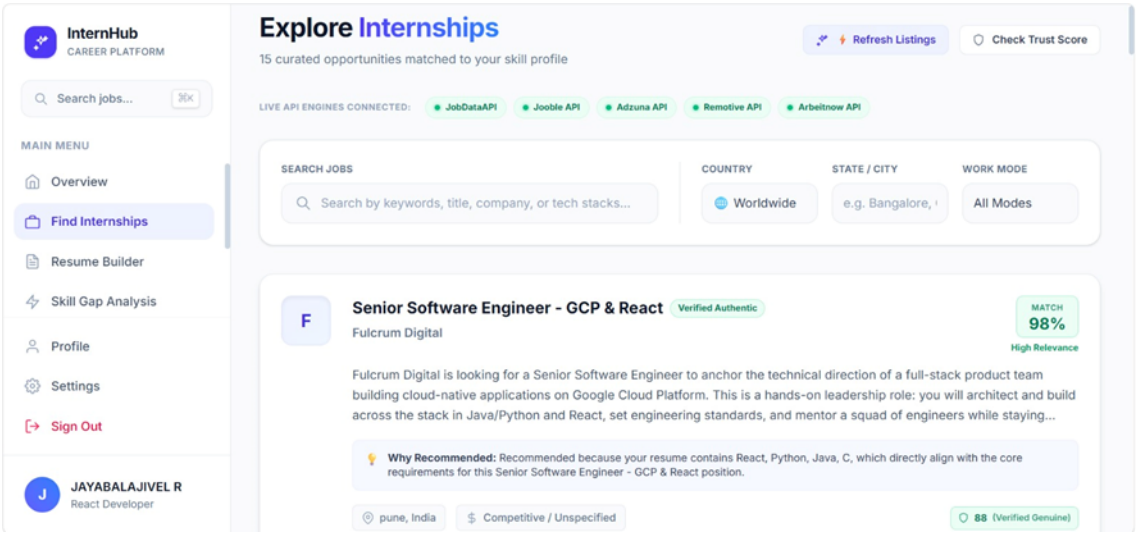


Fig. 4. InternHub internship search and recommendation interface.

The page lists 15 curated opportunities with five connected live API engines and search filters for keywords, country, state/city and work mode.

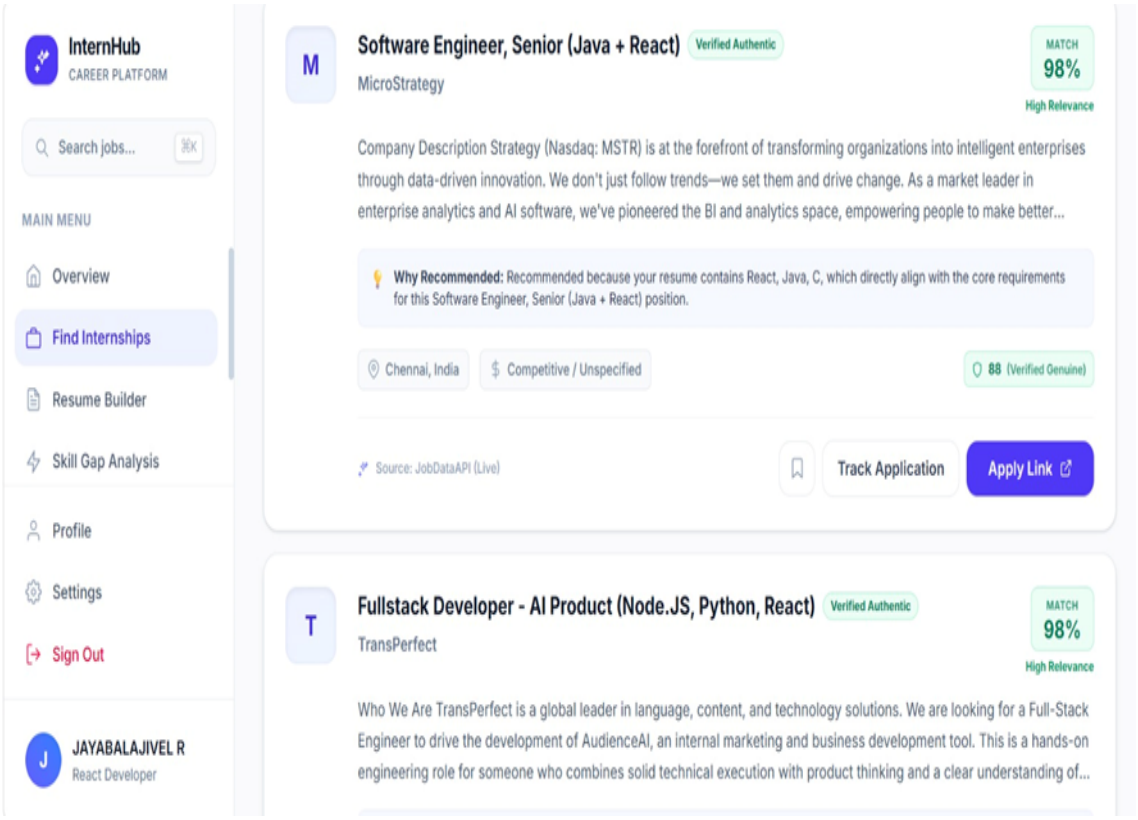


Fig. 5. Recommended internship listings with match percentage and trust score.

Each listing shows a match percentage, a Verified Authentic badge, a Why Recommended note, a trust score of 88, and Track Application and Apply Link actions.

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 X Oct 2026- Available at www.ijraset.com

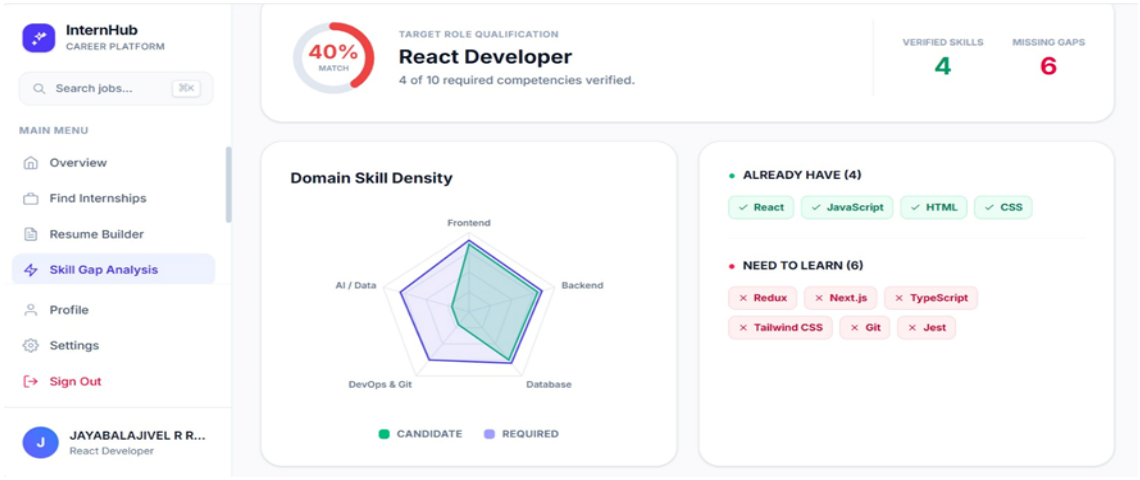


Fig. 6. Skill-gap analysis for the React Developer target role

The page shows a 40% match with 4 of 10 required competencies verified, a candidate-versus-required skill density chart, and lists of skills already held and skills to learn.

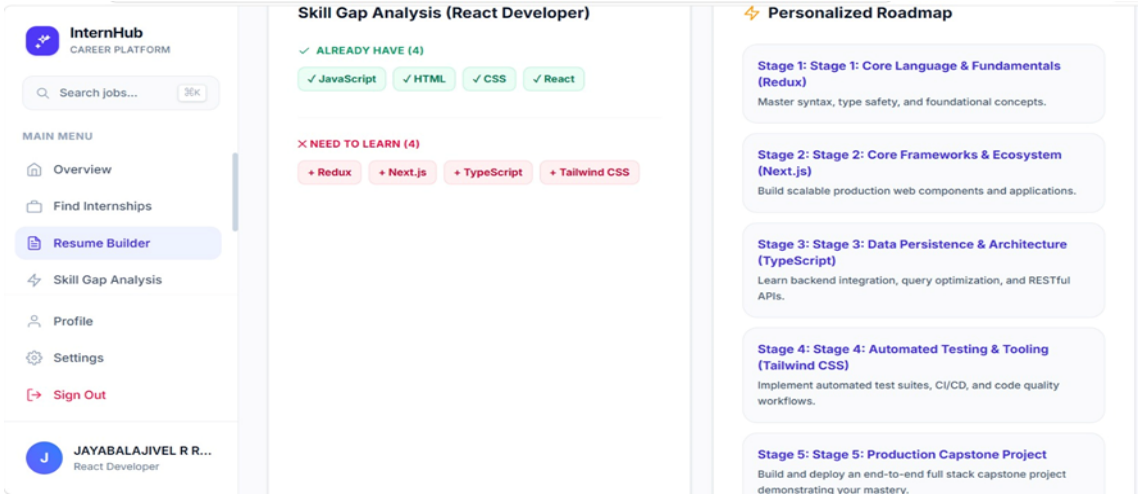


Fig. 7. Personalized learning roadmap generated from the skill gap

Skills already held and skills to learn are listed beside a Personalized Roadmap panel presenting staged learning cards (Stage 1 to Stage 5).

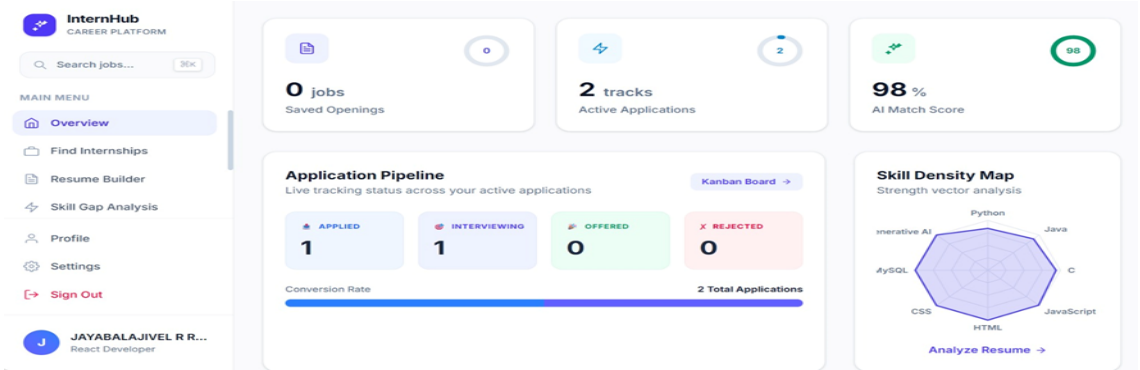


Fig. 8. Overview dashboard with application tracking

The dashboard shows saved openings, active applications and a 98% AI match score, an application pipeline (Applied, Interviewing, Offered, Rejected) and a skill density map.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue X Oct 2026- Available at www.ijraset.com

IV. METHODOLOGY

A. Document Parsing and Negative Heuristic Classification

Resumes of up to 10 MB are parsed with PyMuPDF. Before any semantic processing, the text is compared with 19 financial and billing expressions:

$$P_{bill} = \{tuition\ fee, college\ fee, fee\ receipt, invoice, transaction\ id, amount\ paid, semester\ fee, bank\ challan, \ldots\} \quad (1)$$

A document is classed as *not a resume* if

$$(Count(P_{bill}) \geq 2) \ \vee \ (Count(P_{bill}) \geq 1 \ \wedge \ Count(P_{resume}) = 0) \quad (2)$$

where P_{resume} is a set of expected resume terms. Ingestion then stops and the student sees an explicit error, which keeps unusable records out of the database.

For genuine resumes, the text is divided into education, technical skills and projects. Skills are resolved against a canonical taxonomy with more than 100 aliases (for example *reactjs* and *react.js* map to React, and *k8s* to Kubernetes). Strict word-boundary regular expressions stop false matches such as Java inside JavaScript or C inside Cloud.

B. Provider Aggregation and De-duplication

Postings from all five APIs are requested concurrently with Promise.allSettled(), so one failing provider cannot stop the rest. Each item is mapped to one JSON schema, dead listings are removed by checking the source, and duplicates across providers are resolved with a composite key:

$$K_i = MD5(lower(title_i) \parallel lower(company_i)) \quad (3)$$

C. Trust Score Engine

Every listing starts at 88 points, and deterministic penalties are then applied (Table II). Scores are clamped to the range 15 to 98, and a listing with a score below 65 receives a visible amber warning badge. A listing that demands a security deposit, for example, falls to 48 and is flagged.

TABLE II. TRUST SCORE DEDUCTIONS AND WARNINGS

Detected pattern	Adjustment	Warning shown
Upfront fee or security deposit demanded	−40	Requires upfront registration or financial deposit
Contact through Telegram or WhatsApp only	−40	Directs communication to unverified chat apps
“Typing job” or “earn per day” wording	−40	Suspicious quick-earning or data-entry keywords
URL shortener (bit.ly, t.me)	−25	Application points to an unverified URL shortener
Description under 150 characters	−20	Description is incomplete or lacks qualifications
Link uses http://	−10	Application link lacks SSL/HTTPS encryption

D. Multi-Factor Recommendation Score

Let S be the student's canonical skills, R_i the core skills required by opportunity i and P_i its preferred skills. Three of the five terms are point values computed by fixed rules.

Role alignment, r_i. Take the significant words of the student's primary role (for example *full* and *stack* for Full Stack Developer). The ratio is the number of those words found in the posting title divided by the number of words in the role. A ratio of 0.8 or more gives 20 points, 0.4 or more gives 14, at least one shared word gives 10, and otherwise 0. Dividing by the role, not the title, means a student is not penalised for a long recruiter title such as *Senior Frontend React Software Engineer*.

Project evidence, e_i. Each required skill that also appears in the student's project section adds 4 points, up to 10.

Preferences, π_i. Up to 5 points: 3 if the work mode matches and 2 if the location matches.

The score is

$$M_i = min\left(98, 50 \cdot \frac{|S \cap R_i|}{|R_i|} + 15 \cdot \frac{|S \cap P_i|}{|P_i|} + r_i + e_i + \pi_i\right) \quad (4)$$

If the student shares no required skill with a posting, the score is set to 0, or to at most 15 when the title words strictly overlap, so that an unqualified posting is never recommended. Scores of 70 and above are High Match, 40 to 69 Moderate Match, and below 40 Low Match.

©IJRASET: All Rights are Reserved

TABLE III. WORKED EXAMPLE OF THE FIVE-FACTOR SCORE

Posting	Core (50)	Role (20)	Evidence (10)	Preference (5)	Total	Category
A: Full Stack Intern (React, Node.js, Express.js, MongoDB), remote, Bangalore	50	20	10	5	85	High
B: Data Analyst Intern (Python, SQL, Power BI, Excel), hybrid, Mumbai	25	0	0	0	25	Low
C: Accounts Executive (Tally, GST, Bookkeeping), onsite, Chennai	0	0	0	0	0	Low

The student has Python, SQL, React, Node.js, Express.js, MongoDB, Git, HTML and CSS, project skills in React, Node.js and MongoDB, the target role Full Stack Developer, and a preference for remote work in Bangalore. None of the three postings lists preferred skills. Posting B scores 25 because the student has two of its four required skills, and its title shares no word with the student's role.

E. Lexical Baseline

For comparison, each student profile a and posting b_i is turned into a TF-IDF vector and compared by cosine similarity:

$$TF - IDF(t,d) = TF(t,d) \times \left(ln \frac{1+n}{1+DF(t)} + 1\right) \quad (5)$$

$$cos(a, b_i) = \frac{a \cdot b_i}{\|a\| \, \|b_i\|} \quad (6)$$

For the student and postings of Table III, scikit-learn gives 0.73, 0.20 and 0.00 for postings A, B and C. The order matches the five-factor score, but the cosine value cannot say why a posting ranks where it does.

F. Skill Gap and Learning Roadmap

Missing competencies are the set difference, and the missing skills across the top K opportunities are ranked by how often they appear:

$$G_i = R_i \setminus S, \quad f(s) = |\{i \in Top_K : s \in G_i\}| \quad (7)$$

The platform turns these priorities into a four-phase roadmap: Foundations, Core Frameworks, Advanced Tools and Deployment.

V. EXPERIMENTAL EVALUATION

This section reports three kinds of evidence: component checks on test sets assembled by the authors, timing measurements, and an automated functional suite. They show that the components behave as designed and how fast the system runs. They do not measure how well the ranking agrees with human judgments of relevance, which is left for future work (Section V-E).

A. Environment

All measurements were made on one machine:

- Processor: Intel Core i5-1135G7 at 2.40 GHz (4 cores, 8 threads)
- Memory: 16 GB DDR4
- Storage: 512 GB NVMe SSD
- Operating system: Windows 11 Professional (64-bit)
- Runtimes: Node.js v22.12.0, Python 3.11.9, Chrome 131 (headless)

B. Component Checks

TABLE IV. COMPONENT RESULTS

Component	Test set	Result	Meaning
Non-resume filter	60 documents: 35 resumes in several layouts, 25 non-resumes (tuition receipts, semester challans, hostel bills, tax invoices)	25 of 25 non-resumes rejected, 0 of 35 resumes rejected	No irrelevant file was admitted on this set
Provider de-duplication	500 raw postings from the five feeds	123 removed (24.6%)	About one posting in four was a repeat
Trust Score rules	120 listings: 70 authentic from live feeds, 50 deceptive	94.2% agreement with labels; 47 of 50 deceptive listings flagged (94.0% recall)	Rules fire as designed

Two cautions apply. The 50 deceptive listings were written by the authors, with upfront-payment clauses, chat-only contact, registration-fee traps and typing-job schemes, and the rules were written by the same authors. The 94.2% therefore shows that the rules recognise the patterns they were built for. It is not a detection accuracy on real postings and should not be compared with classifiers evaluated on EMSCAD [8]. Agreement of 94.2% on 120 listings means 113 were classed as labelled, which with 47 deceptive listings caught implies that 4 of the 70 authentic listings received a warning in error. The filter result is likewise based on 60 documents, so it shows that the filter works on typical cases and does not bound its false-rejection rate.

C. Latency

Timing was recorded over 50 consecutive runs and is reported as mean and standard deviation.

TABLE V. RESPONSE TIMES

Stage	Mean	Std. deviation
Full recommendation request, with live API calls	1.84 s	0.22 s
Cached opportunity query	42 ms	6 ms
PDF text extraction (one resume)	310 ms	35 ms
Five-factor scoring of 100 postings	18 ms	3 ms

The full request includes five live API calls, so it depends on provider load and will vary from day to day.

D. Automated Functional Suite

An automated suite of 18 scenarios (src/tests/recommendation.test.ts, 409 lines) covers authentication, the resume pipeline, aggregation, scoring, tracking and access control. All 18 passed (Table VI). On the suite's own fixtures the ranking order matched the expected order (NDCG@3 = 1.0). These cases show that each required behaviour exists. They are chosen by the authors and do not show accuracy on documents the authors did not anticipate.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

TABLE VI. FUNCTIONAL TEST CASES

No.	Module	Scenario	Required behaviour	Result
1	Auth	Register with valid e-mail and password	Account created, profile trigger runs	Pass
2	Auth	Login with wrong password	Access denied with message	Pass
3	Profile	Update domain and location preferences	Changes saved	Pass
4	Resume	Upload a valid technical PDF resume	Skills and target role extracted	Pass
5	Guardrail	Upload a college fee receipt	Rejected with fee alert	Pass
6	Normaliser	Alias tokens (reactjs, k8s, py)	Mapped to React, Kubernetes, Python	Pass
7	Boundary	Java against JavaScript, C against C++	Skills isolated without cross-match	Pass
8	Aggregation	Query live APIs for “intern” in India	More than 35 live listings returned	Pass
9	Tolerance	One external API fails	Remaining APIs still contribute	Pass
10	Dedup	Same job from two APIs	One entry kept	Pass
11	Scam	Listing demands a security deposit	Trust Score below threshold, warning shown	Pass
12	Matching	Web-skilled student, web role	High match (70 or more)	Pass
13	Guardrail	Web-skilled student, accounts role	Zero-match guard applies	Pass
14	Skill gap	Role needs TypeScript and Docker, student lacks both	Both flagged as gaps	Pass
15	Roadmap	Roadmap for Software Engineer	Four-phase milestones shown	Pass
16	Bookmark	Save a recommended listing	Stored in saved_jobs	Pass
17	Tracker	Move an application to Interview	Status and time stamp saved	Pass
18	Security	Student B requests Student A's data	RLS returns zero rows	Pass

E. Protocol for the Remaining Evaluation

Ranking quality will be measured on resumes from several domains. Two reviewers will independently mark each recommended internship as relevant or not and settle disagreements by discussion, with agreement reported as Cohen's kappa. For a student u with relevant set Rel_u,

$$P@K = \frac{|Rel_u \cap Top_K(u)|}{K} \tag{8}$$

Recall@K and mean average precision will be reported with it, and the five-factor score will be compared with keyword search on the job title, a plain count of overlapping skills, and the TF-IDF cosine baseline. The Trust Score rules will be tested on a held-out split of EMSCAD [8], and the non-resume filter on a larger mixed set with false rejections reported separately.

VI. DISCUSSION AND LIMITATIONS

InternHub removes manual profile entry, filters out spam listings and gives an explainable breakdown of every match score. Because the profile comes from a document the student already has, one pass produces a ranked list, a safety indicator for each item and a list of skills to learn. Content-based matching also means the system works on the first day for a new user.

The limitations are as follows.

1) Dictionary-driven skills. Matching depends on the canonical taxonomy. Tools missing from it are not recognised until it is updated, and the taxonomy is built around technical fields, so resumes from accounting or design are described less well.

2) Untuned rules. The Trust Score weights and the 65-point threshold were set by the authors and need tuning on larger labelled sets such as EMSCAD [8]. Some rules can also wrongly penalise a genuine small company, for example a short description or a shortened link.

3) Exact-key de-duplication. The composite key removes identical records but will miss the same job posted under a reworded title.

4) Authored test material. The component test sets and the functional cases were chosen by the authors, so they show that the behaviours exist and not that they hold on unseen data.

©IJRASET: All Rights are Reserved

5) Third-party feeds. Provider rate limits, response times and regional coverage are outside our control and affect both latency and which opportunities appear.

6) Fairness and privacy. Ranking by skill text can disadvantage resumes written in an unusual style, which needs checking on real data. The platform holds personal data, so it should keep only what it needs and let students delete their resume and profile.

VII. CONCLUSION AND FUTURE WORK

InternHub combines resume ingestion, non-resume rejection, multi-provider aggregation, scam screening, explainable five-factor scoring, skill-gap guidance and application tracking in one platform. All 18 functional cases passed, and a full recommendation request averaged 1.84 s with live APIs. The Trust Score and filter results come from test sets built by the authors, so the next step is the evaluation in Section V-E.

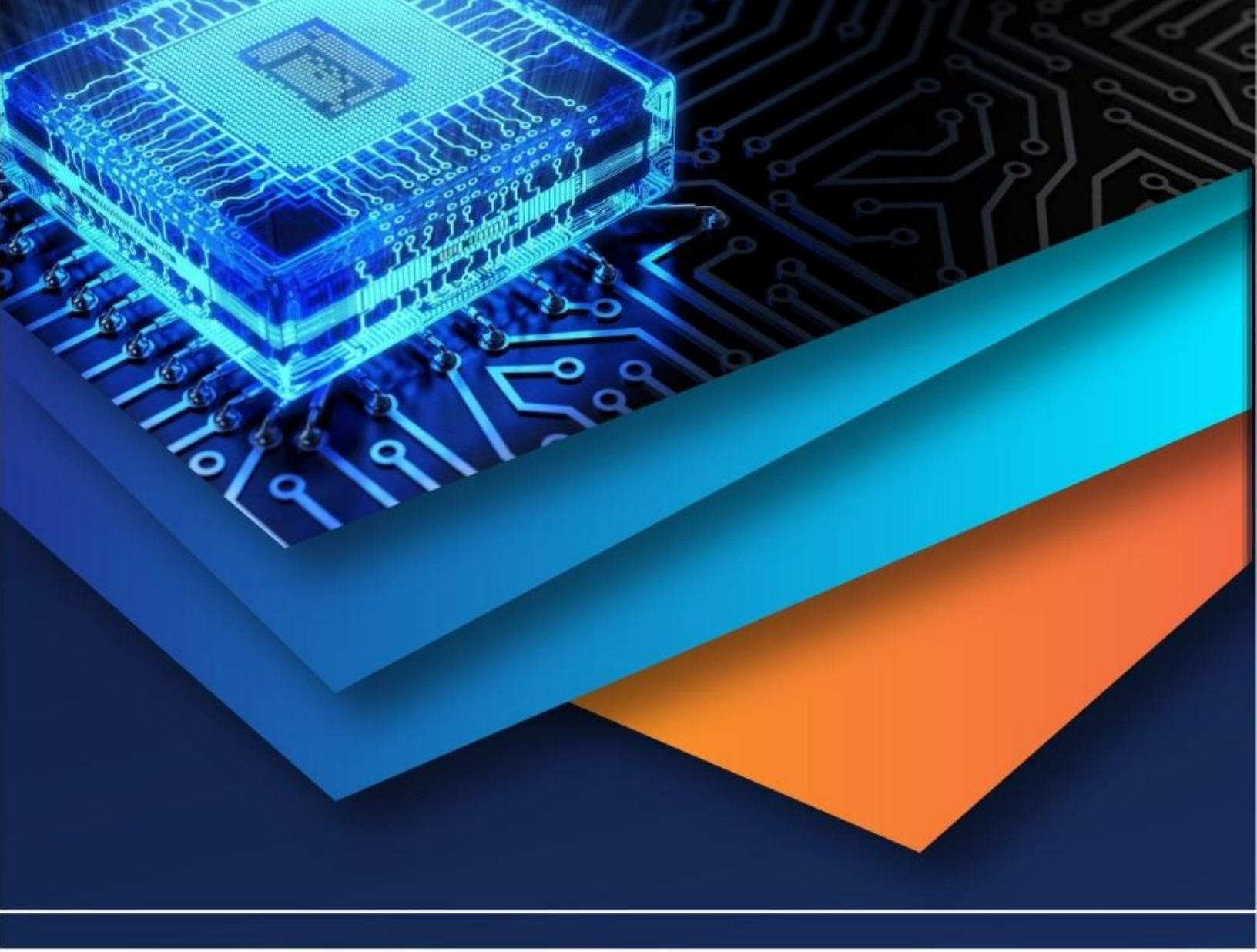
Future work includes Sentence-BERT embeddings for semantic matching, a supervised text classifier trained on EMSCAD to complement the Trust Score rules, a conversational career assistant, and push alerts for opportunities scoring above 85.

VIII. ACKNOWLEDGMENT

The authors thank the Department of Computer Science and Engineering, K.L.N. College of Engineering (Autonomous), Pottapalayam, for the technical infrastructure and facileties provided during this work.

REFERENCES

[1] F. Ricci, L. Rokach, and B. Shapira, Eds., Recommender Systems Handbook, 2nd ed. New York, NY, USA: Springer, 2015.
[2] G. Adomavicius and A. Tuzhilin, “Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions,” IEEE Trans. Knowl. Data Eng., vol. 17, no. 6, pp. 734–749, Jun. 2005.
[3] B. S. Geethanjali, S. Karol, P. M. Reddy, R. V. P. Ganesh, and R. J. Bharathi, “AI-based internship recommendation system,” in Proc. 10th Int. Conf. Smart Struct. Syst. (ICSSS), 2025, doi: 10.1109/ICSSS66939.2025.11346221.
[4] “TechTrack: An internship matching platform for career recommendation using machine learning,” in Proc. 11th Int. Conf. Bus. Ind. Res. (ICBIR), 2026, doi: 10.1109/ICBIR69932.2026.11584730.
[5] S. Mol S. and M. Manoj, “AI-driven internship matching using skill-based recommendation system,” in Proc. Nat. Conf. Emerg. Comput. Appl. (NCECA), vol. 7, no. 1, Amal Jyothi College of Engineering, 2026.
[6] D. R. S. Victoria, L. Gokulraj, R. Hem Kumar, S. Roselin Mary, and S. Nithya Roseline, “AI-based internship recommendation engine for the PM Internship Scheme,” in Proc. Int. Conf. Emerg. Smart Inf. Commun. Technol. (ICESIC), 2026, doi: 10.1109/ICESIC67389.2026.11496549.
[7] A. Busad, G. U. Shankar, S. Dhanush, D. Abbigeri, N. Patil, and K. M. Darshana, “Smart internship recommender,” Int. Res. J. Mod. Eng. Technol. Sci. (IRJMETS), vol. 8, no. 1, pp. 2733–2739, Jan. 2026.
[8] S. Vidros, C. Kolias, G. Kambourakis, and L. Akoglu, “Automatic detection of online recruitment frauds: Characteristics, methods, and a public dataset,” Future Internet, vol. 9, no. 1, art. 6, 2017.
[9] B. Alghamdi and F. Alharby, “An intelligent model for online recruitment fraud detection,” J. Inf. Secur., vol. 10, no. 3, pp. 155–176, 2019.
[10] M. J. Pazzani and D. Billsus, “Content-based recommendation systems,” in The Adaptive Web, P. Brusilovsky, A. Kobsa, and W. Nejdl, Eds. Berlin, Germany: Springer, 2007, pp. 325–341.
[11] P. Lops, M. de Gemmis, and G. Semeraro, “Content-based recommender systems: State of the art and trends,” in Recommender Systems Handbook. Boston, MA, USA: Springer, 2011, pp. 73–105.
[12] G. Salton and C. Buckley, “Term-weighting approaches in automatic text retrieval,” Inf. Process. Manage., vol. 24, no. 5, pp. 513–523, 1988.
[13] K. Spärck Jones, “A statistical interpretation of term specificity and its application in retrieval,” J. Documentation, vol. 28, no. 1, pp. 11–21, 1972.
[14] C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval. Cambridge, U.K.: Cambridge Univ. Press, 2008.
[15] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, “Methods and metrics for cold-start recommendations,” in Proc. 25th ACM SIGIR Conf., 2002, pp. 253–260.
[16] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Item-based collaborative filtering recommendation algorithms,” in Proc. 10th Int. Conf. World Wide Web (WWW), 2001, pp. 285–295.
[17] R. Burke, “Hybrid recommender systems: Survey and experiments,” User Model. User-Adapt. Interact., vol. 12, no. 4, pp. 331–370, 2002.
[18] M. Jones, J. Bradley, and N. Sakimura, “JSON Web Token (JWT),” IETF, RFC 7519, May 2015.
[19] Supabase Documentation. [Online]. Available: https://supabase.com/docs
[20] PostgreSQL Documentation. [Online]. Available: https://www.postgresql.org/docs
[21] Adzuna API Documentation. [Online]. Available: https://developer.adzuna.com
[22] Jooble REST API Documentation. [Online]. Available: https://jooble.org/api/about
[23] Remotive API Documentation. [Online]. Available: https://remotive.com/api
[24] JobDataAPI Documentation. [Online]. Available: https://jobdataapi.com
[25] Arbeitnow Job Board API Documentation. [Online]. Available: https://www.arbeitnow.com/api/job-board-api
[26] FastAPI Documentation. [Online]. Available: https://fastapi.tiangolo.com
[27] spaCy Documentation. [Online]. Available: https://spacy.io
[28] PyMuPDF Documentation. [Online]. Available: https://pymupdf.readthedocs.io
[29] F. Pedregosa et al., “Scikit-learn: Machine learning in Python,” J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)