WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal Homepage: www.ijrpr.com ISSN 2582-7421

AI Multi-Agent Resume Analysis and Career Intelligence System

A. Kanyaga, D. Natchatra, K. Prathipa, Mr. Varun Kumar. B

Assistant Professor, Department of Information Technology, Sri Shakthi Institute of Engineering and Technology, Coimbatore-641062, Tamil Nadu, India.

Email ID: kanyagaa302@gmail.com

ABSTRACT

Job seekers often struggle to understand why their resumes are rejected, which roles genuinely suit their skills, and what they should learn next. Applicant tracking systems (ATS) and recruiters screen resumes quickly, and most candidates receive little structured feedback. This paper presents the AI Multi-Agent Resume Analysis and Career Intelligence System, a web-based application in which a candidate uploads a resume, a pipeline of cooperating software agents extracts skills and experience, evaluates ATS readiness, collects and classifies live job listings, ranks them against the resume, identifies skill gaps, and prepares tailored application material and interview questions. The proposed system aims to combine secure registration and login, resume parsing, explainable resume analysis with severity-rated issues, job description matching, semantic job matching, skill gap analysis with a four-week learning roadmap, generation of a tailored resume, cover letter and HR email, an application tracker, and a consent-based recruiter connection feature. The application uses a Next.js and React frontend, a FastAPI backend, LangGraph for multi-agent orchestration, SQLAlchemy with SQLite or PostgreSQL for data storage, ChromaDB with sentence embeddings for semantic similarity, the Groq API for language-model tasks, and the Jooble API for job listings. Language-model output is treated as assistance, deterministic fallbacks keep the core analysis available when an external service is unavailable, and the generated documents use only facts present in the candidate's resume. The system is designed with consideration for usability, privacy, security, and reliable handling of uploaded resumes. Functional and integration testing can be used to evaluate the individual agents and their interaction. The proposed system provides a foundation for combining resume analysis, job matching, and career guidance in a single application, with further evaluation required on real-world resumes and job data.

Keywords: Multi-Agent System, Resume Analysis, ATS Score, Job Matching, Skill Gap Analysis, LangGraph, Large Language Model, FastAPI, Next.js

1. INTRODUCTION

Finding a suitable job is a decision that depends on how well a candidate's skills, projects, and experience are presented and how closely they match what employers ask for. In practice, a fresher or early-career candidate prepares a resume, searches several job portals, reads long job descriptions, and guesses which skills are missing, often without any feedback on why an application was not shortlisted. This process is time-consuming and difficult to repeat for every role.

The growth of web platforms, large language models, and vector-based text representations has created new possibilities for career services. Language models can read unstructured resume text and return structured fields, embedding models can measure how close a resume is to a job description, and agent frameworks can coordinate several specialised components in a defined order. Combining these technologies may help create a centralized platform that supports candidates with explainable feedback while leaving the final decision to apply, and what to claim, with the candidate.

1.1 Existing challenges in resume screening and job search

Existing approaches include job portals, generic resume templates, online ATS checkers, and advice from mentors. Although these channels make jobs easy to find, they usually treat resume checking, job search, and skill planning as separate activities, and many ATS checkers return a single score without explaining what should be changed. Candidates therefore often need to use several tools and still cannot tell which missing skill or weak section affects their chances most.

Another challenge is the dependence on keyword and format knowledge. Skill names appear in several spellings, bullet points may start with weak verbs or lack measurable results, dates may be written inconsistently, and tables or text boxes may not be read correctly by screening software. Proprietary ATS scores are also not publicly documented, so any tool can provide only a transparent estimate and cannot predict the decision of a real screening system.

1.2 Role of multi-agent AI and transparent scoring

A multi-agent design divides the problem into smaller tasks, each handled by an agent with a clear input and output: reading the resume, collecting jobs, classifying domains, matching, finding skill gaps, writing application material, and preparing interview questions. A coordinator passes a shared state from one agent to the next, which keeps each step easy to test and replace. Language models are used where free-form reading or writing is needed, such as parsing a resume or drafting a cover letter.

Matching and scoring can be handled by transparent, rule-based formulas. A job's match score combines semantic similarity, skill overlap, experience, ATS readiness, and education with documented weights, and a resume score is calculated from contact details, section completeness, skills, length, and the severity of detected issues. However, AI outputs may contain errors, and results should be presented as assistance together with the reasons behind each score.

1.3 Problem statement and objectives

Many existing tools keep resume analysis, job search, skill planning, and application preparation separate. This makes it difficult for candidates to see how a change in their resume affects the jobs they can target. There is therefore a need for an integrated application that provides convenient access to analysis, matching, and guidance while considering accuracy, transparency, privacy, and ease of use.

The main objective of this work is to propose a web-based AI Multi-Agent Resume Analysis and Career Intelligence System that combines resume parsing, ATS-oriented analysis, job collection and matching, skill gap analysis, and application support in one platform. The system aims to explain every score and suggestion, and to reduce the effort needed to shortlist suitable roles and prepare for them.

The specific objectives are:

i. To provide secure registration and login for candidates using hashed passwords and token-based sessions.

ii. To accept resumes in PDF or text form with validation and extract skills, experience, education, and projects.

iii. To calculate an ATS-oriented resume score and list severity-rated issues with suggested fixes.

iv. To compare a resume with a pasted job description and report matched and missing keywords.

v. To collect job listings, classify them into technology domains, and rank them using semantic and skill-based scoring.

vi. To identify missing skills for a target domain and generate a four-week learning roadmap.

vii. To generate a tailored resume, cover letter, HR email, and interview preparation questions, and to track applications.

viii. To evaluate the application through functional, integration, and usability testing.

2. MATERIALS AND METHODS

2.1 Development approach

The project follows a structured software development approach consisting of problem identification, requirement analysis, system design, implementation, integration, testing, and evaluation. During problem identification, the main challenges associated with unexplained resume rejection, scattered job search, and unclear skill planning are considered.

Requirement analysis identifies the major functions required by the application, including authentication, resume upload, resume analysis, job collection, matching, skill gap analysis, application generation, and tracking. System design defines the relationship between the user interface, REST API, agent pipeline, language-model and embedding services, and data storage components.

The implementation stage develops the frontend pages and backend modules using Next.js, React, and FastAPI. Integration connects the individual agents with the Groq API, the Jooble job service, ChromaDB, and the relational database so that they can exchange information correctly. Testing is then used to identify errors, examine the application's behaviour, and determine areas requiring improvement.

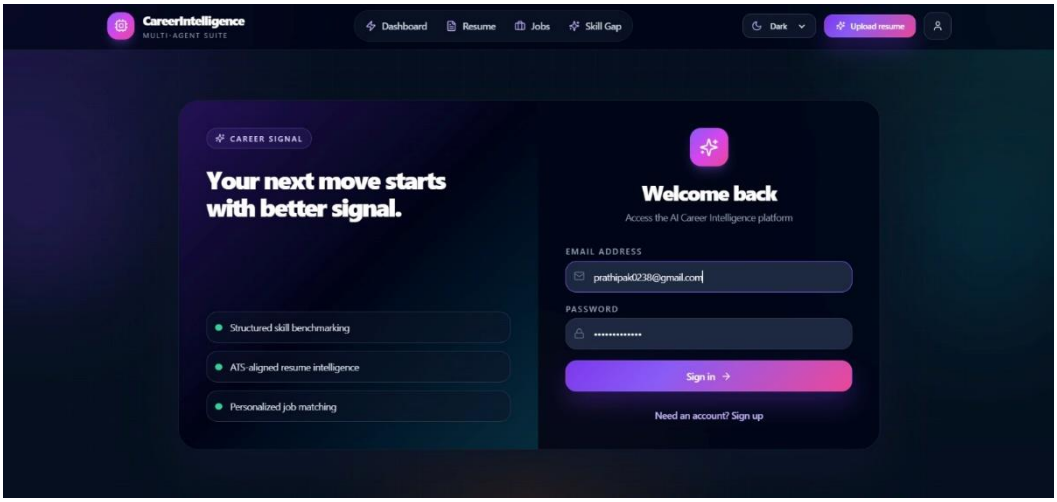
2.2 System architecture

The proposed system is organized into multiple layers, including the user interface layer, application logic layer, authentication layer, multi-agent intelligence layer, and data storage layer.

The user interface layer provides the screens through which users register, log in, upload resumes, view analysis, browse matched jobs, generate applications, and chat with the career advisor. The application logic layer is a FastAPI REST API that processes requests and coordinates communication between the interface and backend services. The authentication layer issues JSON Web Tokens and protects the routes that return personal data, while

the multi-agent layer, orchestrated by a LangGraph coordinator, manages resume intelligence, job collection, domain classification, matching, skill gap analysis, application generation, and interview preparation.

The data storage layer uses a relational database, SQLite by default with PostgreSQL supported through the connection setting, for users, resumes, jobs, match results, applications, notes, career goals, and bookmarks, and ChromaDB for vector embeddings of jobs and resumes. The architecture is intended to keep the modules logically separate so that individual components, such as the skill gap agent or the matching engine, can be maintained and improved without redesigning the complete application. The login screen of the implemented application is shown below.



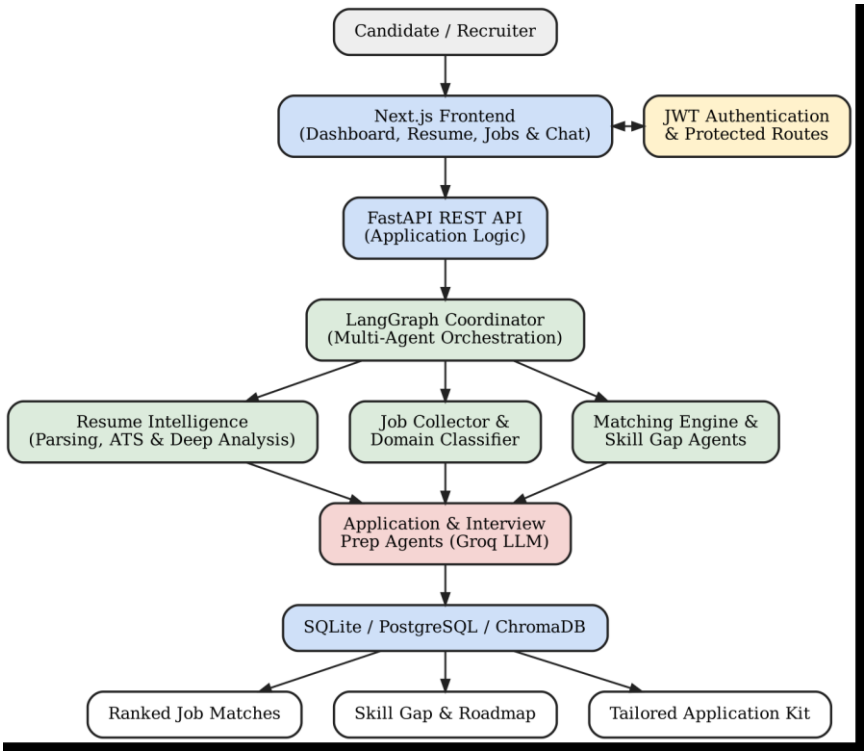


Fig. 1. System Architecture of the Proposed AI Multi-Agent Resume Analysis and Career Intelligence System.

2.3 Technology stack and development environment

The web-based implementation uses Next.js 14 with React and TypeScript for page navigation, Tailwind CSS for styling, Framer Motion for transitions, and Axios for API communication. The interface includes pages for login, dashboard, resume analysis, jobs, skill gap, application generation and tracking, career chat, templates, recruiter connect, settings, and an administrator view. The backend uses Python with FastAPI to expose REST endpoints, with dependency-based authentication for protected routes.

SQLAlchemy with asynchronous drivers stores the application records. ChromaDB stores job and resume embeddings generated with the all-MiniLM-L6-v2 sentence-embedding model [4], [5]. LangGraph defines the agent workflow as a graph of nodes sharing a typed state [2], and the Groq API provides

the language model used for resume parsing, bullet and summary improvement, application writing, and interview questions [3]. Job listings are obtained from the Jooble API [10]. Passwords are hashed with bcrypt, and logged-in sessions use JSON Web Tokens [6], [7].

Development and testing can be performed using an appropriate code editor, Python, Node.js, and a supported web browser. The required environment depends on the frontend framework, backend runtime, external services, and API credentials, which are supplied to the backend through environment variables and are not stored in the source code.

2.4 Data collection and processing

The system processes information required to provide its features, such as user account details, uploaded resume files and their extracted text, parsed skills, experience, education and projects, resume scores, job listings, match results, generated application documents, application notes, and career goals. Job listings are normalized, de-duplicated, and indexed before they are matched.

Personal information should be collected only when necessary for the application's intended functions. Resumes contain sensitive personal information, so they are linked to the owning user account, and the data of one user is not returned to another user. The recruiter connection feature shares a candidate profile only when the candidate has given explicit visibility consent.

The application validates inputs before processing them and handles errors such as unsupported file types, empty files, files larger than 10 MB, password-protected or unreadable PDFs, resumes with too little text, and unavailable AI or job services. Uploaded resumes are accepted only as PDF or TXT files. The prompts that generate application material treat resume and job text as untrusted data so that text inside a document cannot change the task, and they instruct the model to use only facts present in the resume.

2.5 Database design

The database is organized into relational tables. A suitable data model contains records for users, resumes, jobs, match results, applications, application notes, career goals, bookmarks, and recruiter profiles and connections.

User records maintain the name, email, hashed password, and target role. Resume records store the file name, extracted text, target domain, parsed skills, experience, education and projects, ATS score, resume score, and summary. Job records store the title, company, location, salary range, domain, description, extracted skills, source, and application link. Match records associate a user, a resume, and a job with the overall, semantic, skill, experience, and ATS scores, the match reasons, and the missing skills.

Application records store the tailored resume, cover letter, and HR email for a job, with a status that can be tracked from draft to applied, interview, offer, or rejection, and application notes can carry interview dates. Career goal records store the target role, target domain, skills to learn, milestones, and projects. The database should be configured to prevent unauthorized access to personal data, and only the information required for the application should be retained.

2.6 Resume analysis, matching and skill gap logic

The analysis process begins when a candidate uploads a resume. The backend extracts the text, and the Resume Intelligence agent sends it to the Groq language model, which returns skills, experience, education, projects, and a summary in structured form [3]. When the language-model service is not configured or fails, a deterministic fallback keeps the analysis available. A deep analysis module then checks the text for weak action verbs, bullet points without measurable results, inconsistent date formats, missing contact details or sections, and formatting patterns such as tables, text boxes, and dot leaders that may not be read by ATS software. Each issue is given a severity of critical, high, medium, or low and a suggested fix.

The resume score is a weighted combination of five components: the issue score (50%), section completeness (20%), contact completeness (10%), number of skills (10%), and length (10%). The issue score starts at 100 and is reduced by 20, 10, 5, or 2 points for each critical, high, medium, or low issue. For a pasted job description, the job description matching module extracts technology keywords and reports the percentage of required keywords present in the resume, together with the matched and missing keywords. This is a transparent keyword estimate and is not the proprietary score of any real ATS.

The matching engine converts the resume and each job description into embedding vectors and computes their cosine similarity. For each job it also calculates the overlap between the resume skills and the skills extracted from the job. The overall match score is the weighted sum summarised in Table 1, limited to the range 50 to 99, and each job is returned with readable match reasons and its missing skills [5]. Jobs are sorted by score. The experience and education factors currently use fixed reference values and are intended to be refined with parsed experience and education details.

The skill gap agent compares the normalized resume skills with the skills expected for the target domain, such as Full Stack, Frontend, Backend, AI/ML, DevOps, or Cyber Security. The skill gap score is the percentage of expected skills present, the remaining skills are listed as missing, and a four-week learning roadmap of foundations, applied practice, integration, and review is built from those missing skills. The personalized application agent then drafts a tailored resume, cover letter, and HR email for a selected job, and the career recommendation agent prepares technical, coding, and behavioural interview questions with model answers. Automated analysis and generated documents should complement, rather than replace, the candidate's own review, and the application should communicate these limitations to the user.

Scoring component	Weight (%)	Basis of calculation
Semantic similarity	35	Cosine similarity between the resume embedding and the job text embedding, scaled to 0–100
Skill match	30	Share of the job's extracted skills found in the resume, plus a 20-point base, limited to 100
Experience	15	Experience factor; currently a fixed reference value pending parsed-experience scoring
ATS readiness	10	ATS-oriented score of the uploaded resume
Education	10	Education factor; currently a fixed reference value pending parsed-education scoring
Total	100	Weighted sum, rounded and limited to the range 50–99

Table 1. Match-score components used by the job matching engine.

2.7 Multi-agent analysis workflow

The workflow begins when a candidate registers, logs in, and uploads a resume. The application validates the file type, size, and content before extracting the text. The LangGraph coordinator then runs the agents in a fixed order: resume intelligence, job collection, domain classification, matching, skill gap analysis, personalized application, and career recommendation, with each agent reading from and writing to a shared state.

The Resume Intelligence agent stores the parsed skills and scores. The Job Collector agent obtains listings from the Jooble service, normalizes and de-duplicates them, and indexes them in ChromaDB, and the Domain Classifier agent assigns each job to a technology domain. The Matching Engine scores and ranks the jobs, and the Skill Gap agent produces the missing skills and roadmap. The results are stored so that the dashboard, jobs page, and skill gap page can display them.

After processing, the application should provide clear feedback about the status of the analysis. If a step fails, the coordinator returns a clear error message instead of partial or invented results. The candidate can then bookmark jobs, generate a tailored application for a selected job, update its status in the application tracker, ask the career chat advisor for guidance, and choose to make a profile visible to other users through recruiter connect.

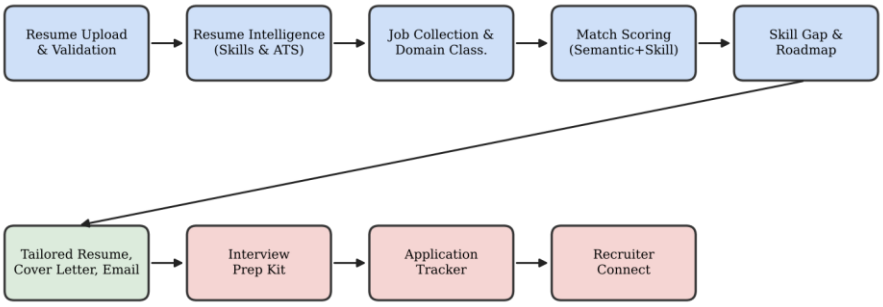


Fig. 2. Workflow of the Proposed AI Multi-Agent Resume Analysis and Career Intelligence System.

2.8 Testing and evaluation

Testing is required to evaluate the correctness and reliability of the proposed system. Unit testing can be used to examine individual functions such as cosine similarity, domain classification, skill normalization, and scoring, while integration testing checks communication between the user interface, REST API, authentication, the database, ChromaDB, the Groq service, and the Jooble service.

System testing evaluates the overall application workflow, including registration, login, resume upload, analysis, job matching, skill gap analysis, application generation, tracking, and recruiter connect. User interface testing examines readability, navigation, responsiveness, and ease of understanding of the displayed results.

The evaluation should also include error conditions such as invalid credentials, unauthorized access, unsupported or oversized files, empty or unreadable resumes, expired tokens, and unavailable AI or job services. Any quantitative claims about parsing accuracy, match quality, or response time should be reported only when supported by actual measurements.

3. RESULTS AND DISCUSSION

3.1 Proposed system functionality

The proposed application brings several career-related functions into a centralized interface. These include secure authentication, resume upload and parsing, ATS-oriented scoring with explainable issues, job description matching, semantic job matching, skill gap analysis with a learning roadmap, tailored application generation, interview preparation, an application tracker, a career chat advisor, and consent-based recruiter connect.

The integration of these functions aims to reduce the effort required to move between job portals, resume checkers, and planning tools. The actual performance of the application must be established through testing of the implemented features and their interactions.

3.2 Functional evaluation

Functional evaluation examines whether the implemented modules behave according to their requirements. Representative test scenarios for each module are summarised in Table 2. The results should be documented based on the observed behaviour of the application, and features that have not been tested should be clearly identified as pending evaluation.

Test ID	Module	Test scenario	Expected behaviour
T1	Authentication	Register and log in with valid and invalid credentials	Valid user receives a token; invalid user rejected with a message
T2	Access control	Call a protected endpoint without a valid token	Request denied with an authorization error
T3	Resume upload	Upload an unsupported, empty, oversized, or unreadable file	File rejected with a clear validation error
T4	Resume analysis	Upload a text-based resume	Skills extracted; ATS score and severity-rated issues returned
T5	JD matching	Paste a job description for a stored resume	Matched and missing keywords with a coverage score
T6	Job matching	Run matching on collected jobs	Jobs returned ranked by match score with reasons and missing skills
T7	Application	Generate a tailored application and update its status	Resume, cover letter, and HR email saved; status shown in the tracker

Table 2. Representative functional test scenarios for the proposed system.

Any failed tests should be investigated and repeated after corrections are made. The resume parsing module in particular should be evaluated on a separate set of labelled resumes, and its accuracy reported per field rather than only as a single overall figure. The matching module should be evaluated using jobs of known suitability.

3.3 Comparison with conventional approaches

Conventional approaches may require candidates to check their resume on one website, search jobs on another, and plan their learning separately. The proposed system aims to bring resume analysis, job matching, skill gap analysis, and application support together in a common interface, with an explanation for each score and recommendation, potentially improving clarity and accessibility.

The inclusion of language-model based parsing and writing provides an opportunity to reduce manual effort. However, the usefulness of this approach depends on resume quality and the verified performance of the model, and the candidate remains responsible for checking every generated document. The proposed system should not be considered more reliable than existing services without comparative testing.

3.4 Discussion

The proposed architecture provides a framework for integrating resume analysis, job matching, and personalised career guidance. A centralized interface can make the effect of resume changes easier to understand, while modular agents can simplify maintenance and the addition of new domains, skills, or scoring rules.

The system also faces practical challenges. Resume parsing may misread unusual layouts, scanned PDFs, or multi-column designs, and keyword-based skill matching may miss related terms. The experience and education factors of the match score are currently fixed reference values, and the skill lists for each domain are curated rather than learned from the market. Job listings depend on the availability and coverage of the external job service, and the language-model service requires network access and an API key, so deterministic fallbacks are used when these are unavailable.

Privacy and security are important because the application handles personal resume data. Suitable authentication, authorization, password hashing, protection of secrets through environment variables, restricted cross-origin access, consent for profile visibility, and safe handling of uploaded files are therefore necessary [11]. Further evaluation should measure parsing accuracy, match relevance, response time, and the usability of the application. These measurements would provide stronger evidence of the system's effectiveness in practical situations.

4. CONCLUSION

The AI Multi-Agent Resume Analysis and Career Intelligence System presents a technology-based approach to helping candidates understand their resumes, find suitable jobs, and plan their learning through a centralized application. The proposed system combines resume parsing, explainable scoring, job matching, skill gap analysis, and application support, with scope for additional features such as notifications and richer analytics.

The modular multi-agent architecture provides a foundation for integrating these functions while considering usability, privacy, and data security. However, the practical effectiveness of the application depends on the correct implementation and testing of its individual agents, the quality of the submitted resumes, and the accuracy of the language model and job data.

Future development can focus on parsing experience and education for the match score, supporting more domains and learned skill lists, handling scanned resumes through text recognition, adding resume export in standard templates, introducing notifications and messaging between connected users, and improving ranking with feedback from candidate behaviour. Further testing and user evaluation are required to determine the reliability of the system on real-world resumes.

In conclusion, the proposed application offers a foundation for developing an integrated resume analysis and career intelligence system. With appropriate implementation, validation, and privacy safeguards, it can contribute to making job preparation more transparent and accessible for candidates.

5. ACKNOWLEDGMENTS

The authors would like to thank Sri Shakthi Institute of Engineering and Technology for providing the academic environment and facilities required for this project. The authors also express their sincere gratitude to the Head of the Department of Information Technology, the project guide, and the faculty members for their guidance, encouragement, and support throughout the development of the project.

6. REFERENCES

1. S. Ramírez. FastAPI Documentation. Available at: https://fastapi.tiangolo.com/
2. LangChain. LangGraph Documentation. Available at: https://langchain-ai.github.io/langgraph/
3. Groq. GroqCloud API Documentation. Available at: https://console.groq.com/docs
4. Chroma. ChromaDB Documentation. Available at: https://docs.trychroma.com/
5. N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in Proc. EMNLP-IJCNLP, 2019, pp. 3982–3992.
6. M. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," RFC 7519, Internet Engineering Task Force, 2015.
7. N. Provos and D. Mazières, "A Future-Adaptable Password Scheme," in Proc. USENIX Annual Technical Conference, 1999.
8. SQLAlchemy. SQLAlchemy Documentation. Available at: https://docs.sqlalchemy.org/
9. Vercel. Next.js Documentation. Available at: https://nextjs.org/docs
10. Jooble. Jooble REST API Documentation. Available at: https://jooble.org/api/about
11. OWASP Foundation. OWASP Application Security Guidance. Available at: https://owasp.org/