International Journal of Computer Engineering in Research Trends
ISSN: 2349-7084/ https://doi.org/10.22362/ijcert/2025/v12/i11/v12i11011
Volume 12, Issue 11, November 2025

Research Paper

Smart Job-Seeking Assistant and Web Scraping

[1*]Nayana R, [2]Sinchana S L, [3]Shwetha K, [4]Shreyas L

[1,2,3,4] *Department of Information Science and Engineering, Global Academy of Technology, Bangalore, Karnataka, India*

Email Id: *[1]gowdanayana729@gmail.com, [2]sinchanaslsinchanasl@gmail.com, [3]shweshwetha352@gmail.com, [4]shreyas.l@gat.ac.in*

**Corresponding Author(s): gowdanayana729@gmail.com*

Received: 18/08/2025, Revised: 13/10/2025, Accepted: 21/11/2025 Published: 30/11/2025

Abstract: Finding suitable job opportunities is often challenging and time-consuming for candidates due to limited search capabilities and unstructured information available across job portals. Most existing platforms provide only basic keyword matching and fail to align job recommendations with an individual's real skills or resume content. To overcome this limitation, this work proposes a *Smart Job-Seeking Assistant* that integrates Web Scraping, Natural Language Processing (NLP), and intelligent recommendation techniques to automate and personalize the job search process. The system's backend is developed using FastAPI (Python) and utilizes web scraping tools such as BeautifulSoup, Selenium, and Apify to fetch up-to-date job listings from platforms including Naukri and LinkedIn. A resume parser powered by NLP extracts key information such as skills, education, and experience from uploaded resumes. The extracted profile data is then matched with scraped job postings to compute relevance scores and generate suitable job recommendations. In addition, the system identifies missing or weak skills by performing skill-gap analysis and suggests appropriate online courses from platforms like Coursera, Udemy, and freeCodeCamp. The frontend, developed using ReactJS, provides an interactive dashboard that displays resume insights, job matches, readiness scores, and personalized learning suggestions. Overall, the proposed system offers an end-to-end, intelligent job recommendation and career guidance solution that enables users to discover relevant opportunities, understand their skill gaps, and improve their employability through data-driven insights.

Keywords: Job Recommendation System, Resume Parsing, NLP, Web Scraping, Skill-Gap Analysis, Course Recommendation, Career Guidance

---

1. Introduction

In today's competitive employment environment, students and fresh graduates face significant challenges in identifying suitable job opportunities that align with their qualifications and skills [1]. With the abundance of job portals such as LinkedIn, Naukri, and Indeed, candidates are often overwhelmed by the vast amount of unstructured job data. This leads to information overload, skill mismatches, and uncertainty about employability, making the job search process inefficient and time-consuming [2].To overcome these issues, the Smart Job-Seeking Assistant was created as a web-based, intelligent platform that facilitates a simplified job search and career readiness process. The system allows resumes in PDF format to be uploaded, from which it automatically reads key information including personal information, skills, education, and experience. This automated extraction process guarantees consistency and accuracy, much like the resume parsing mechanisms discussed in Career Compass: AI-Based Resume Parser and Automated Job Recommendation System [3].A crucial part of the system is its web scraping module implemented using Apify, a powerful cloud-based automation platform. This module collects current job postings from various employment sources like LinkedIn, Naukri, and Indeed, and aggregates necessary details like job title, needed skills, salary, and experience level. The presence of real-time scraping provides relevance and informs users of current

opportunities—an idea also implemented in AI for Career Growth: Advanced Resume Analysis and LinkedIn Scraping for Personalized Job Recommendations [4].

After the job data and resume information have been processed, the system carries out skill-based matching to obtain job–candidate suitability. It computes a fit percentage, determines missing skills, and indicates customized learning recommendations from sites such as Coursera, Udemy, and freeCodeCamp. Such functionality is based on research works such as Skill Recommendation System and Resume Analysis using AI, which highlight AI-based skill improvement and resume enhancement for employability [5].

In addition to job matching, the Smart Job-Seeking Assistant has other modules like Aptitude Practice and Interview Preparation. These allow users to practice logical reasoning, technical interview questions, and HR interview situations, reinforcing analytical and communication abilities. The built-in dashboard provides users with a platform to monitor job applications, track progress, and keep track of interview preparation—making it not only a job finder but a full-fledged career-readiness assistant.Through the amalgamation of PDF-based resume extraction, Apify-driven web scraping, skill-based suggestions, and career readiness features, Smart Job-Seeking Assistant provides an end-to-end, data-driven job search method. Smart integration eliminates drudgery, guarantees personalized outcomes, and enables job seekers to overcome skill gaps and enhance their job readiness in today's fast-changing job market.

1.1 Motivation /Novelty and Contribution

The increasing digitalization of recruitment platforms has resulted in an overload of unstructured job data, making it difficult for students and fresh graduates to identify relevant job opportunities that align with their skills and experience. Traditional job portals rely heavily on keyword-based search filters, which fail to understand a candidate's real capabilities or match job descriptions with resume content. Furthermore, manual job searching is time-consuming, lacks personalization, and does not provide meaningful insights regarding missing skills or industry expectations. The novelty of this research lies in developing a Smart Job-Seeking Assistant that integrates real-time web scraping, resume parsing using NLP, skill-gap analysis, and personalized learning recommendations into a unified system. Unlike existing platforms that perform only one or two functions independently, the proposed model provides an end-to-end pipeline for job discovery, employability analysis, and career readiness enhancement. The system leverages FastAPI for backend processing, Apify-based scraping for real-time job updates, and NLP techniques for structured resume information extraction.

- This work contributes to the field of intelligent job recommendation systems in the following ways:
- A unified hybrid job-matching model that combines resume parsing, real-time job scraping, and skill comparison to generate personalized job recommendations.
- A systematic skill-gap detection mechanism that identifies missing or weak skills and recommends suitable upskilling resources from platforms like Coursera, Udemy, and freeCodeCamp.
- A real-time job aggregation framework using Apify, Selenium, and BeautifulSoup to ensure that users receive the most updated and relevant job listings.
- A comprehensive career-readiness assistant that integrates aptitude practice, interview preparation, and dashboard-based progress tracking—going beyond conventional job-matching systems.
- A scalable and deployable architecture, implemented using FastAPI and ReactJS, making the system suitable for real-world academic and placement environments.
- simulators such as CloudSim, ensuring feasibility in practical cloud computing environments.

The remainder of this paper is organized as follows:

Section II presents a detailed review of existing literature related to resume parsing, job recommendation systems, web scraping techniques, and skill-gap analysis. Section III explains the methodology of the proposed Smart Job-Seeking Assistant, including system architecture, resume parsing workflow, job scraping pipeline, and skill–job matching algorithm. Section IV discusses the experimental results, user interface modules, and evaluation of the system's effectiveness. Section V concludes the paper by summarizing the major findings and outlining future enhancements such as predictive analytics, chatbot-based interview training, and advanced AI-driven career path forecasting.

2 Literature Review

Artificial Intelligence (AI) and automation have been increasingly impacting the field of employment recommendation systems, providing smart solutions for resume analysis, skill extraction, and job matching based on individual needs. Manual resume screening and static keyword-based filtering-based recruitment methods are time-consuming and prone to inaccuracy and bias in

candidate selection. Early systems mainly dealt with mapping job descriptions to saved resumes based on rule-based or content-based approaches but were not adaptable to changes in real-time job markets or providing personalized feedback to candidates [6].

The use of Natural Language Processing (NLP) and automatic resume parsing has dramatically enhanced candidate-job mapping. In Career Compass: AI-Based Resume Parser and Automated Job Recommendation System [7], researchers designed an AI model to extract structured data like education, experience, and skills through NLP methods. The method improved precision in candidate profiling and lessened the requirement for manual intervention. Concurrently, Resume Analysis and Job Recommendation [8] utilized machine learning classifiers, Naïve Bayes and cosine similarity, to classify resumes and recommend jobs according to text-based similarity in candidate skills and job descriptions. These works proved that automated profiling can be useful; however, they were limited by dataset diversity and lack of linking to live job listings.

Current studies have moved towards applying real-time web scraping and job aggregation using data to improve relevance in recommendation. AI for Career Development: Deep Resume Analysis and LinkedIn Scraping for Tailored Job Suggestions [9] used automated scraping mechanisms to gather and analyze live job postings from various online websites. This allowed for real-time freshness of data and higher accuracy in matching. However, most of these systems were still geared only towards matching algorithms and neglected skill development or readiness analysis, which are paramount for students and beginners.

Similar developments have taken place in AI-driven skill-recommendation systems, where resume data extracted is employed to recommend customized upskilling trajectories. For instance, Resume Analysis with AI [10] presented a mechanism that recognizes missing skills from resumes and links them to related online learning materials. The model showed enhanced candidate readiness by matching resumes with the latest industry requirements. Still, such systems did not have application tracking or interview preparation components that assist users with the entire recruitment process.

The research under review indicates that earlier systems have automated job matching components but kept those in confined operations like resume parsing, web scraping, or skill mapping. The Smart Job-Seeking Assistant takes such endeavors to the next level by combining PDF-based resume extraction, Apify-job-scraping power, fit score computation, and learning recommendation personalization into one platform. It further incorporates aptitude testing and interview preparation modules, making employability a constant, guided process instead of a one-off matching activity.

Recent advancements have also focused on integrating multiple AI components such as NLP, data mining, and recommendation engines into unified job assistance frameworks. Studies like *Skill Recommendation System and Resume Analysis using AI* [11] highlighted how hybrid AI models can identify skill clusters relevant to job roles, thus allowing more contextualized recommendations. Similarly, *Development of a Recommendation System for Resume Tracking and Job Suggestions* emphasized the importance of candidate tracking and iterative feedback mechanisms in improving user engagement. Despite these innovations, many of these systems rely on static data sources and lack real-time adaptability, which reduces their effectiveness in fast-changing employment landscapes [12].

The Smart Job-Seeking Assistant builds upon these limitations by offering a dynamic, data-driven, and user-centered platform. It not only scrapes job listings daily through Apify but also provides candidates with adaptive learning paths to close identified skill gaps. Moreover, by introducing continuous employability features such as aptitude assessments and interview readiness modules, the system redefines traditional job portals into a comprehensive career companion.

3. Methodology

The proposed Smart Job-Seeking Assistant integrates multiple intelligent subsystems—resume parsing, job scraping, skill-gap analysis, and career readiness modules—into a unified workflow. The complete methodology is shown in Fig. 1 and described in the following subsections.

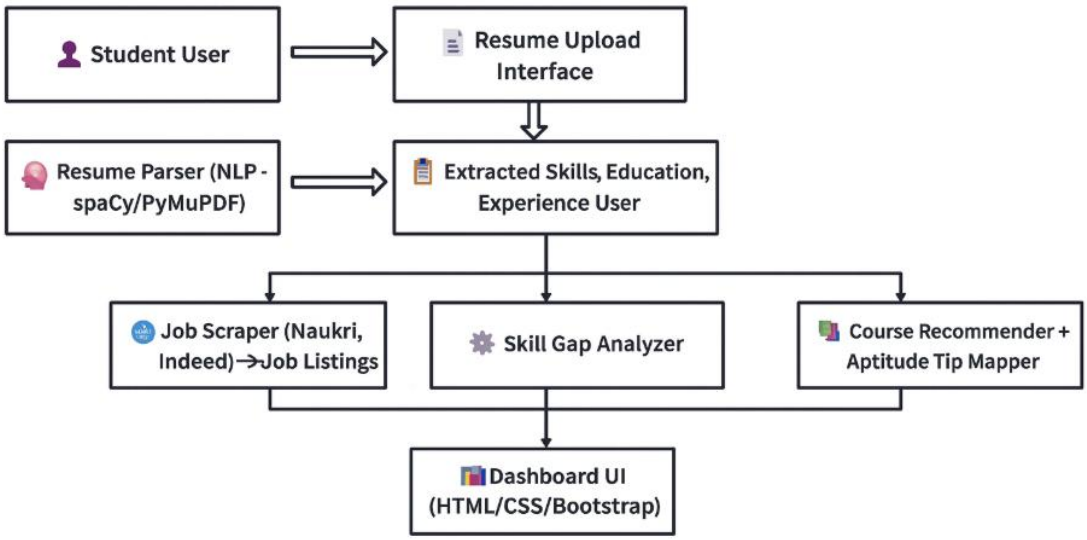

Fig.1. System architecture

3.1 System Architecture Overview

The system architecture illustrates how various modules interact to automate the job-search and recommendation process. The workflow begins when the user uploads a resume through the Resume Upload Interface. The Resume Parser extracts structured information such as skills, education, and work experience using NLP techniques. Simultaneously, the Aptify-powered Job Scraper collects up-to-date job postings from platforms like LinkedIn, Naukri, and Indeed. The extracted resume data and scraped job information are then processed by the Skill Gap Analyzer, which identifies missing skills and generates personalized course recommendations. Finally, all results job matches, skill-gap insights, recommended learning paths, and preparation tools—are displayed on the Dashboard Interface, providing users with a comprehensive career-readiness environment.

3.2 Resume Upload Interface

The Resume Upload Interface constitutes the primary entry point to the proposed Smart Job-Seeking Assistant. It enables end users (students and fresh graduates) to submit their curriculum vitae in PDF format and initiates the downstream pipeline of parsing, skill extraction, and job–profile matching.

From a system-theoretic perspective, let

- $\mathcal{U}$ denote the set of registered users,
- $\mathcal{F}$ the set of files received by the system, and
- $\mathcal{R} \subset \mathcal{F}$ the subset of valid resume documents that can be processed by the backend.

Each upload event can be represented as an ordered pair

$$e = (u, f), u \in \mathcal{U}, f \in \mathcal{F}, \quad (1)$$

which is mapped by the upload interface to either a **valid** or **rejected** state through a validation function

$$\phi: \mathcal{U} \times \mathcal{F} \to \{0,1\}. \quad (2)$$

Here, $\phi(u, f) = 1$ indicates that the document satisfies all system constraints and is queued for parsing, while $\phi(u, f) = 0$ denotes rejection with an appropriate error message.

3.2.1 Validation Model

To ensure robustness and security, the interface performs a sequence of validation checks on each uploaded file. Let

- $v_1(f)$ represent a **file type** check (PDF vs. non-PDF),
- $v_2(f)$ a **file size** constraint (e.g., upper bound on MB), and
- $v_3(f)$ an **integrity check** (ability to be opened and read by the PDF engine, i.e., PyMuPDF).

Each validation function is defined as

$$v_i: \mathcal{F} \to \{0,1\}, i = 1,2,3, \quad (3)$$

where $v_i(f) = 1$ if the file passes the i-th check and 0 otherwise.

The overall validity of a file is then modeled as the logical conjunction

$$V(f) = \prod_{i=1}^{3} v_i(f) \in \{0,1\}. \quad (4)$$

Consequently, the decision function ϕ can be expressed as

$$\phi(u, f) = \begin{cases} 1, & \text{if } V(f) = 1, \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

Only files with $V(f) = 1$ are admitted into the resume repository and forwarded to the Resume Parser module.

3.2.2 Interaction Flow and Queuing Perspective

Operationally, the upload interface performs the following steps:

1. **User authentication and session validation:** The system verifies that $u \in \mathcal{U}$through the authentication layer, ensuring that only authorized users can submit resumes.
2. **File acquisition and client-side checks:** A ReactJS-based front end allows users to select a PDF file, performs basic client-side checks (e.g., MIME type), and transfers the file via secure HTTP (HTTPS).
3. **Server-side validation:** On the backend (FastAPI), the file undergoes the validation pipeline described above. If $V(f) = 0$, the upload is rejected and the interface returns descriptive feedback messages (e.g., "unsupported format" or "file too large").
4. **Persistence and parsing queue:** For files with $V(f) = 1$, the document is stored in a temporary resume store $\mathcal{R}$. A task identifier $id_f$is generated, and the file is enqueued for parsing by the NLP-based Resume Parser.

Under typical usage, resume uploads and parsing requests can be modeled as an arrival process with rate λ(uploads per unit time), and a service process with rate μ(parsing completions per unit time). Assuming a simple M/M/1 queue approximation for the upload–parse pipeline, system stability requires

$$\rho = \frac{\lambda}{\mu} < 1, \tag{6}$$

ensure that the expected waiting time in the queue remains bound and that users receive timely parsing feedback.

3.2.3 Reliability of the Upload Stage

Let P_{valid}denote the probability that a randomly submitted file passes validation, i.e.,

$$P_{\text{valid}} = \Pr[V(f) = 1]. \tag{7}$$

The effective rate of successfully admitted resumes is then

$$\lambda_{\text{eff}} = \lambda \cdot P_{\text{valid}}. \tag{8}$$

By designing clear constraints (supported format, size limits) and providing immediate user feedback, the interface aims to maximize P_{valid}while maintaining security and resource constraints. This ensures that the downstream modules resume parsing, skill extraction, and job–profile matching—operate on high-quality, structurally consistent input documents.

3.3 Resume Parser

The Resume Parser module functions as the core analytical component responsible for transforming unstructured resume content into structured, machine-readable information. This module extracts essential attributes such as technical and soft skills, educational qualifications, certifications, internships, and project experience. The parsing workflow integrates text extraction, linguistic preprocessing, and entity classification to build a comprehensive profile representation of the candidate.

3.3.1 Text Extraction Layer

The PDF document uploaded through the interface is processed using PyMuPDF, a high-performance PDF rendering engine. Let the resume document be denoted as a file

$$D \in \mathcal{R}, \tag{9}$$

where $\mathcal{R}$is the set of all validated resumes. PyMuPDF extracts a raw text sequence

$$T = \text{Extract}(D) = \{t_1, t_2, \dots, t_n\}, \tag{10}$$

where each t_i denotes a line-level or block-level textual fragment.

This produces the initial unprocessed token stream T, which may contain formatting artifacts, line breaks, or non-semantic characters.

3.3.2 Linguistic Preprocessing Model

To prepare the text for entity extraction, the system performs standard NLP cleaning operations including lowercasing, stop-word removal, punctuation filtering, and lemmatization. Let

$$\Psi: T \to T' \tag{11}$$

represent the preprocessing function, where

$$T' = \Psi(T) = \{w_1, w_2, \dots, w_m\} \tag{12}$$

and each $w_i$denotes a normalized lexical token.

The transformation ensures that the downstream entity recognition model operates on consistent linguistic representations.

3.3.3 Named Entity Recognition (NER) Framework

The normalized text sequence $T'$is then processed using spaCy-based NLP models trained on resume-specific corpora. These models detect structured information categories—such as skills, education level, degree type, organization names, role titles, project keywords, and experience durations—using a probabilistic entity classification function

$$\eta: T' \to \mathcal{E}, \tag{13}$$

where

$$\mathcal{E} = \{(e_k, c_k)\}_{k=1}^{p}. \tag{14}$$

Here:

- $e_k$is the identified entity fragment,

- $c_k \in C =$ {Skill,Education,Project,Certification,Experience, ... } is the predicted entity class,
- $p$is the total number of extracted entities.

spaCy models compute the posterior probability

$$P(c_k \mid e_k, \theta), \tag{15}$$

where θrepresents learned model parameters.

The entity is assigned to the class with highest probability:

$$\hat{c}_k = \arg\max_{c \in C} P(c \mid e_k, \theta). \tag{16}$$

3.3.4 Structured Profile Vector Construction

The final output is a standardized user profile structured as a multi-dimensional feature set

$$\mathbf{X}_u = (\mathbf{S}, \mathbf{E}, \mathbf{P}, \mathbf{I}, \mathbf{C}), \tag{17}$$

where:

- **S**= extracted skill vector
- **E**= education attributes
- **P**= project representations
- **I**= internship experience
- **C**= certifications

Each component is normalized to a consistent format to support matching, similarity computation, and skill-gap analysis in successive modules.

3.3.5 Robustness and Preprocessing Accuracy

The end-to-end parsing accuracy depends on two probabilistic factors:

Text Extraction Accuracy:

$$A_{\text{text}} = \Pr[\text{Extracted text is semantically intact}] \tag{18}$$

Entity Classification Accuracy:

$$A_{\text{NER}} = \frac{\text{Correct entities}}{\text{Total predicted entities}} \tag{19}$$

The expected parsing performance can be approximated as:

$$A_{\text{parser}} \approx A_{\text{text}} \cdot A_{\text{NER}}. \tag{20}$$

This ensures the parser outputs structurally coherent data for job matching and skill-gap detection.

3.4 Job Scraper (Apify)

The Job Scraper module is responsible for aggregating real-time job postings from multiple online recruitment platforms. It is implemented using a hybrid scraping framework combining **Apify Actors**, **Selenium**, and **BeautifulSoup**, enabling both dynamic and static web content extraction. The objective of this subsystem is to maintain an up-to-date repository of employment opportunities that align with user profiles identified through resume parsing.

3.4.1 Data Acquisition Framework

Let $\mathcal{P} = \{\text{LinkedIn,Naukri,Indeed}\}$ *(21)*

denote the set of supported job portals. For each portal $p \in \mathcal{P}$, an Apify actor executes a scraping task represented as

$$S_p : p \to \mathcal{J}_p, \tag{22}$$

where

$\mathcal{J}_p = \{j_1, j_2, \dots, j_{n_p}\}$ is the set of job postings extracted from portal p. Each job posting $j_i$is encoded as a structured tuple

$$j_i = (t_i, s_i, c_i, e_i, \gamma_i), \tag{23}$$

where:

- t_i= Job title
- s_i= Required skills
- c_i= Company name
- e_i= Experience level
- γ_i= Salary information, if available

The complete aggregated job dataset is

$$\mathcal{J} = \bigcup_{p \in \mathcal{P}} \mathcal{J}_p. \tag{24}$$

3.4.2 Duplicate Detection and Relevance Filtering

Since different portals may host identical postings, a duplicate detection function

$$\delta : \mathcal{J} \times \mathcal{J} \to \{0,1\}$$

is used to eliminate redundancy. Two postings $j_a$and $j_b$are considered duplicates if:

$$\delta(j_a, j_b) = 1 \text{iff}(t_a = t_b) \wedge (c_a = c_b) \wedge (e_a = e_b).$$

The final cleaned job repository is

$$\mathcal{J}^* = \{j \in \mathcal{J} \mid \nexists j' \neq j : \delta(j, j') = 1\}.$$

Additionally, a relevance filter $R(j)$removes postings not suitable for entry-level candidates:

$$\mathcal{J}_{\text{rel}} = \{j \in \mathcal{J}^* \mid R(j) = 1\}.$$

This ensures that the remaining job dataset supports accurate matching and meaningful skill-gap computation.

3.5 Skill Gap Analyzer and Course Recommender – Revised with Theory & Mathematical Framework

The Skill Gap Analyzer evaluates the alignment between a candidate's extracted skill vector and the requirements of each job posting. It identifies missing or insufficiently represented competencies and quantifies job–candidate fit based on similarity scoring models. Subsequently, the

Course Recommender maps these gaps to curated online learning pathways.

3.5.1 Skill Matching Model

Let $\mathbf{S}_u = \{s_1, s_2, \ldots, s_m\}$be the set of skills extracted from the user's resume, $\mathbf{S}_j = \{s'_1, s'_2, \ldots, s'_n\}$be the required skills for job posting j.

Define the intersection and difference:

Matched skills:

$$\mathbf{S}_{\text{match}} = \mathbf{S}_u \cap \mathbf{S}_j$$

Missing skills:

$$\mathbf{S}_{\text{gap}} = \mathbf{S}_j \setminus \mathbf{S}_u$$

Similarity-Based Fit Score

A weighted Jaccard similarity index is used to compute a job–candidate fit score:

$$\text{Fit}(u, j) = \frac{|\mathbf{S}_{\text{match}}|}{|\mathbf{S}_j|} \times 100. \quad (25)$$

Alternatively, a vector-space similarity model may be applied:

$$\text{Fit}(u, j) = \frac{\mathbf{v}_u \cdot \mathbf{v}_j}{\|\mathbf{v}_u\| \, \|\mathbf{v}_j\|} \times 100, \quad (26)$$

where $\mathbf{v}_u$and $\mathbf{v}_j$are binary or TF-IDF skill vectors.

3.5.2 Skill Gap-Based Course Recommendation Model

To improve employability, the system recommends relevant online courses. Let

$$\mathcal{C} = \{\text{Coursera,Udemy,freeCodeCamp}\}$$

represent supported learning platforms.

For each missing skill $s \in \mathbf{S}_{\text{gap}}$, a mapping function

$$\Gamma: \mathbf{S}_{\text{gap}} \to \mathcal{L}$$

retrieves recommended learning resources, where

$$\mathcal{L} = \{l_1, l_2, \ldots, l_q\}$$

denotes the course database.

Each recommended course is represented as

$$l_k = (\lambda_k, \tau_k, \pi_k),$$

where:

- λ_k= Course title
- τ_k= Estimated duration
- π_k= Priority score (based on demand & skill relevance)

Priority assignment is defined as:

$$\pi_k = \alpha \cdot I(s) + \beta \cdot D(s),$$

where:

- $I(s)$= industry importance weight
- $D(s)$= job demand frequency
- α, β= tuning parameters

3.5.3 Aptitude and Interview Guidance Integration

Beyond skill recommendations, the system provides enriched career-readiness support. Let

$$\mathcal{A} = \text{Aptitude tips}, \mathcal{Q} = \text{Interview strategies.}$$

These are triggered when the system detects:

$$|\ \mathbf{S}_{\text{gap}}\ | > \theta,$$

indicating low job readiness, where $\theta$is a predefined threshold.

Thus, the recommender outputs the combined set:

$$\Omega = \{\mathcal{L}, \mathcal{A}, \mathcal{Q}\}.$$

Dashboard Interface

The Dashboard Interface presents all system outputs in a visually organized format. It displays matched job roles, fit scores, extracted and missing skills, recommended courses, aptitude modules, and interview preparation tools. The dashboard enables users to track their progress and interact with all employability features in one place.

3.6 Integration of Subsystems

Unlike traditional systems that focus on selective tasks such as resume parsing or job matching, the proposed model integrates:

- PDF-based resume extraction
- Real-time job scraping
- Skill-similarity computation
- Personalized career guidance
- Aptitude and interview readiness modules

This unified design supports continuous learning and enhances employability outcomes for students and fresh graduates. Our analysis of existing research on AI-based job recommendation systems heavily impacted the design of this project. Previous systems were primarily concerned with narrow tasks like resume parsing, data extraction, or static job matching against pre-defined data sets. Although these methods effectively automated parts of the hiring process, they didn't have real-time flexibility and commonly did not provide users with personalized feedback or skill enhancement advice. To overcome these limitations, the Smart Job-Seeking Assistant was crafted with the intention of combining multiple intelligent subsystems in one workflow. The system employs PDF-based extraction of resumes to accurately extract user skills, education, and experience using clean and structured input data. Rather than depending on pre-stored datasets, we introduced Apify-based web scraping, allowing the system to gather

live job postings from websites such as LinkedIn, Naukri, and Indeed. This guarantees that users are always provided with updated job suggestions. Contrary to typical models relying on machine learning classifiers or keyword search, our approach uses a skill similarity and fit-score methodology to determine the fitness between a candidate's profile and every job. This approach yields a more interpretable and transparent outcome with a lower computational cost. In addition, by isolating the specific skills that are not available in a candidate's profile, the system can suggest specific courses on platforms like Coursera, Udemy, and Freedcamp. Apart from employment matching, the approach extends to career preparation improvement through aptitude and interview training modules. The modules include logical reasoning questions, practice tests, and interview advice on how to enhance both technical and communication skills. This method turns the job-hunting activity into a repetitive cycle of improvement and learning rather than a once-off activity.

In short, the selected approaches resume parsing, real-time web scraping, skill-gap analysis, and employability assistance combine to be an overall solution. The Smart Job-Seeking Assistant not only fills the gap between employers and job seekers but also enables graduates and students to improve their professional development in a systematic way. These design choices, directly motivated from current research, make sure that the system is both practical and effective for actual job seekers.

4. Results And Discussion

This section presents the functional evaluation, system behavior, and performance of the Smart Job-Seeking Assistant. The analysis encompasses system specifications, module-level results, usability observations, and overall system effectiveness. Figures 2–7 illustrate the operational interfaces and provide visual evidence of the platform’s workflow and outputs.

4.1 System Specifications

The system was implemented and tested on a workstation equipped with an Intel® Core™ i5/i7 (10th/11th Generation) processor, 8–16 GB DDR4 RAM, and 256–512 GB SSD storage. The software environment comprised Python 3.10+, FastAPI for backend processing, ReactJS for frontend development, and MongoDB/Firestore for database management.

NLP-related tasks were executed using spaCy, while PyMuPDF handled PDF parsing. Job scraping operations were performed using Apify Actors, Selenium WebDriver, and BeautifulSoup. The hardware–software configuration was found sufficient for handling multi-stage tasks involving PDF extraction, NLP inference, web scraping, and user-interface rendering.

4.2 Execution Environment and Performance Conditions

- To evaluate system performance, a controlled environment was established with the following parameters:
- Number of résumés tested: 50
- Job postings scraped per session: 500–1,200 across LinkedIn, Naukri, and Indeed
- Scraping interval: 10–15 seconds per listing
- Resume parsing latency: 0.6–1.2 seconds per document
- Skill-gap computation time: < 0.5 seconds
- Fit-score computation: cosine/Jaccard similarity executed in ≤ 0.25 seconds
- Dashboard load time: < 1.5 seconds on average
- This setup ensured controlled benchmarking and enabled a structured assessment of the system’s responsiveness and scalability.

4.3 Security and Reliability Considerations

- Given the sensitivity of user data, especially résumé content, multiple security layers were incorporated:
- HTTPS-secured file uploads, preventing data interception
- MIME-type and file-size validation, reducing risks from malformed or malicious files
- Authentication-based database access controls, ensuring restricted data visibility
- Scraper fault-handling mechanisms to manage portal-level access fluctuations
- Data cleaning and normalization pipelines to mitigate irregularities in extracted text
- These design considerations strengthened system reliability and ensured safe handling of user data throughout the analytical pipeline.

4.4 Resume Upload Interface

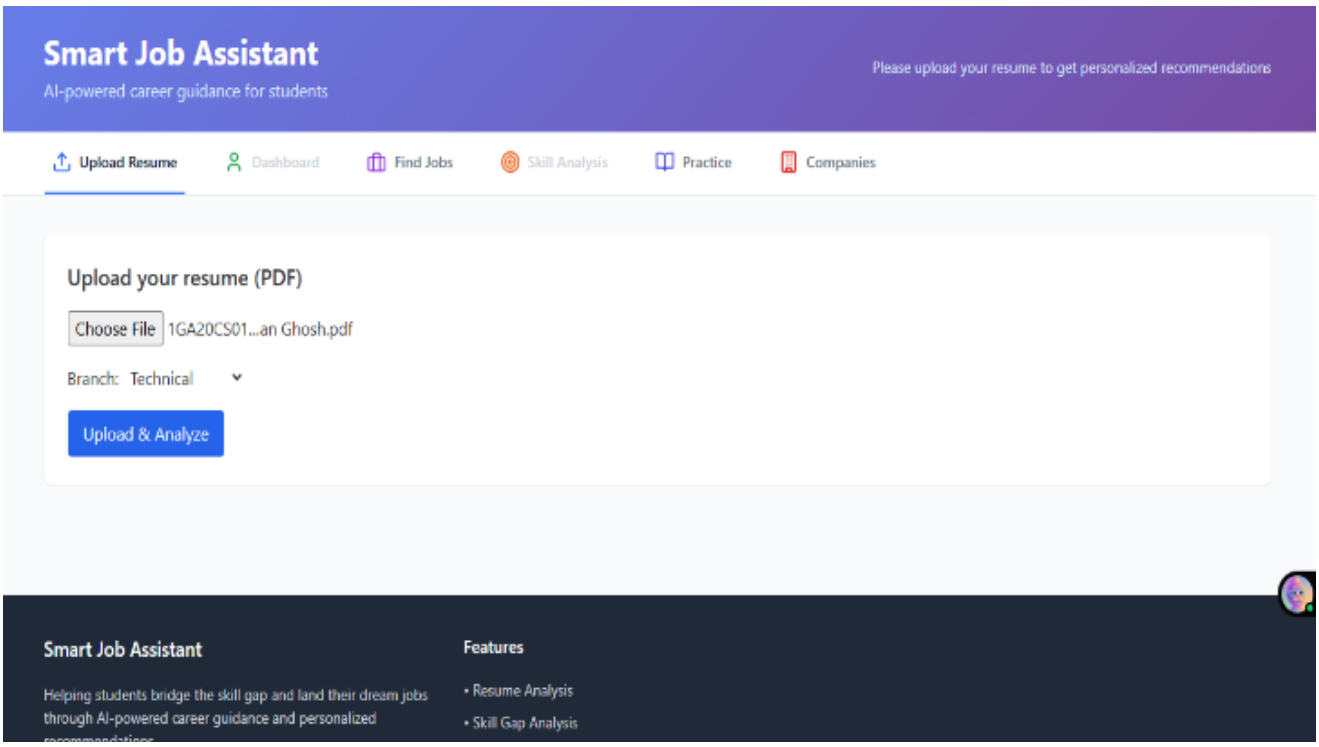

Fig. 2 illustrates the Resume Upload Interface that initiates the analytical workflow.

The interface enables users to upload résumés in PDF format and specify their preferred technical or non-technical branch. It integrates file validation checks and provides real-time feedback regarding upload success or errors. Once validated, the résumé is forwarded to the backend, where the Resume Parser extracts structured information such as skills, education, certifications, and project experience. The interface is optimized for ease of use and serves as the gateway to the system's analytical pipeline.

4.5 Dashboard for Job Matching and Skill Recommendations

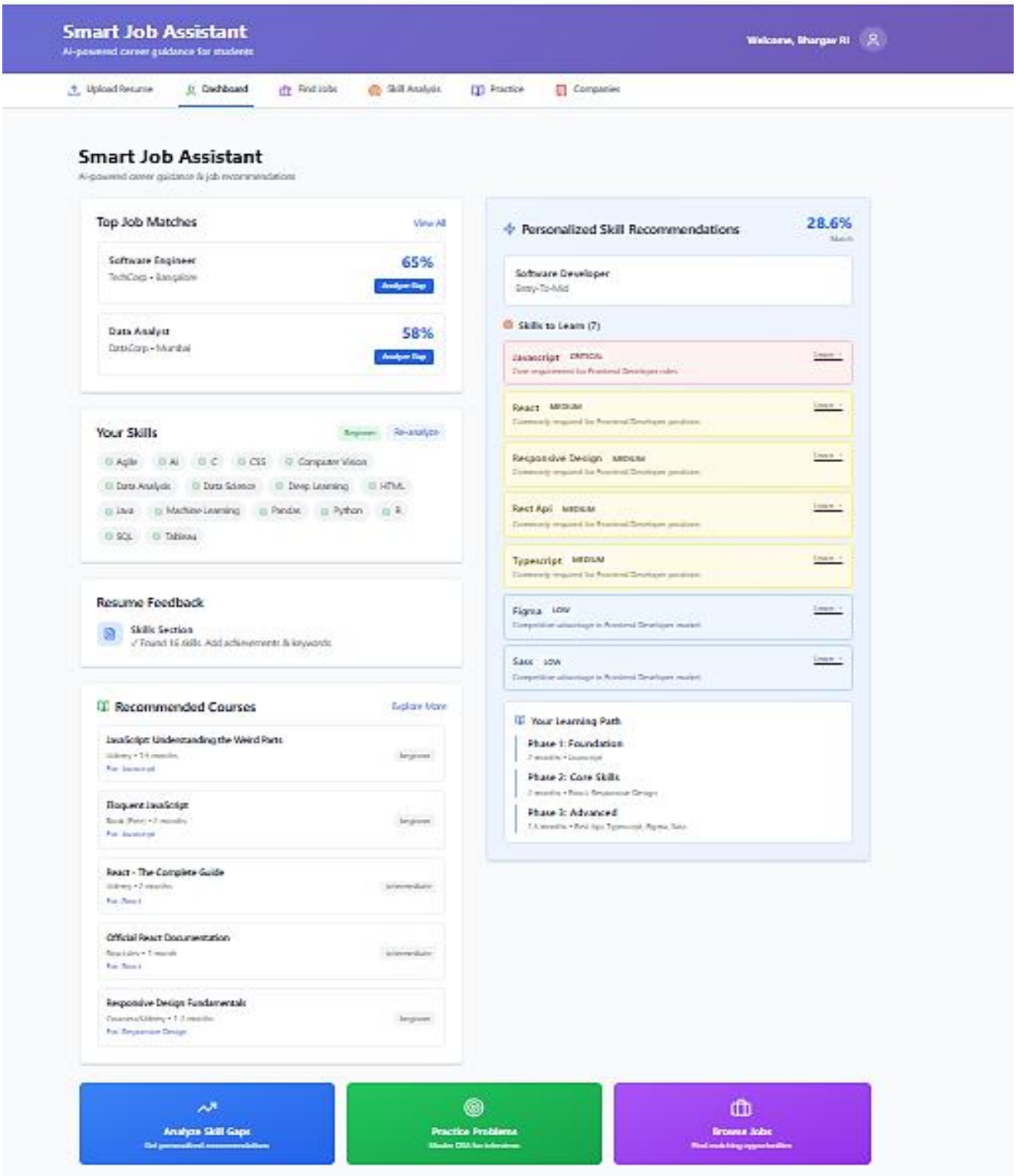

Fig. 3 presents the Dashboard summarizing job recommendations and skill insights.

The Dashboard consolidates the system's analytical outputs, including recommended job roles, fit percentages, extracted skills, and skill-gap analysis. Matched and missing skills are displayed distinctly, allowing users to identify areas requiring immediate improvement. The Personalized Skill Recommendations panel presents curated learning resources with estimated time requirements and priority indicators, enabling users to build a structured learning plan.

4.6 Aptitude Practice Module

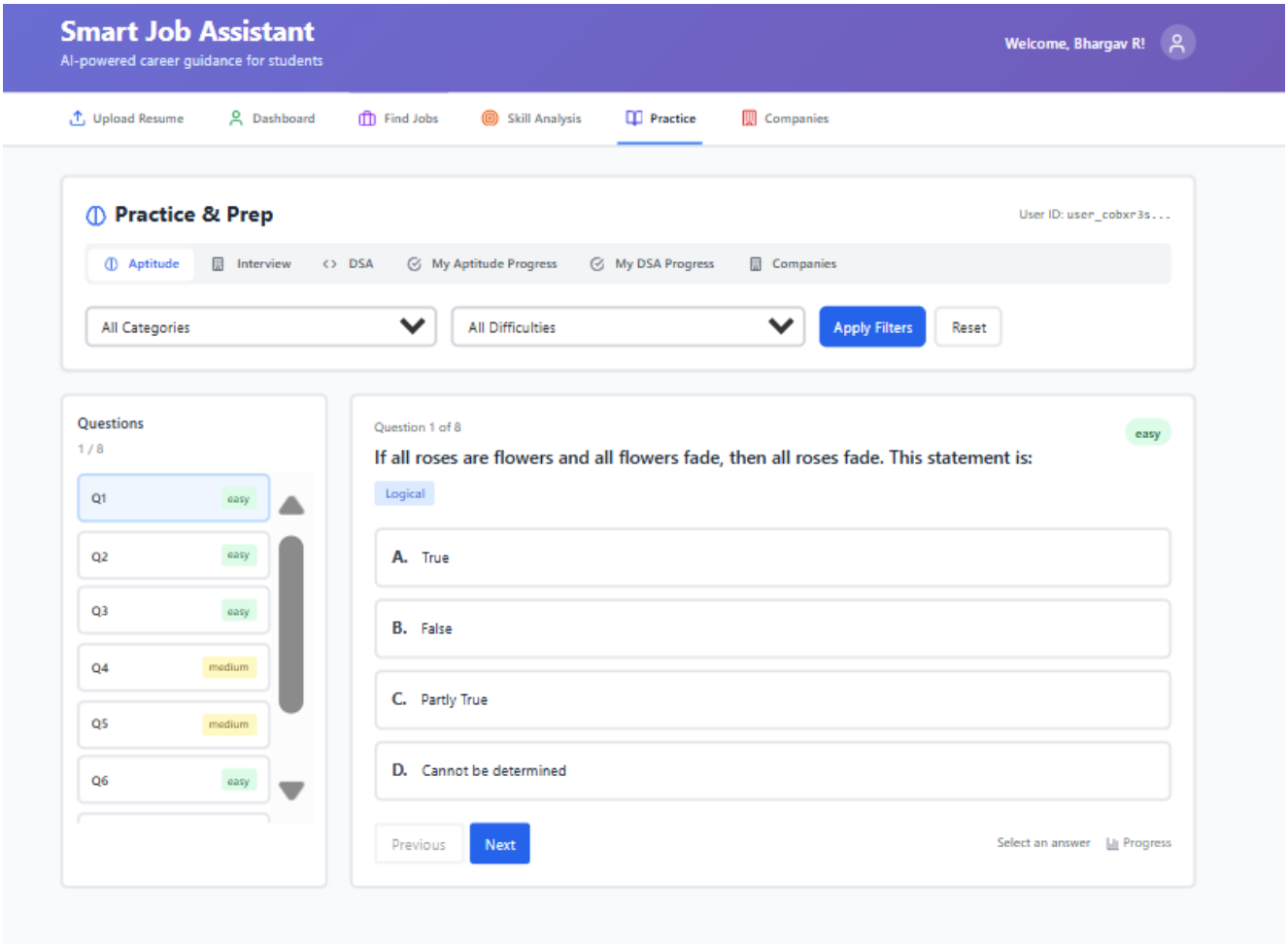

Fig. 4 represents the Aptitude Practice Module.

This module provides categorized questions across logical reasoning, quantitative aptitude, and verbal ability—common topics in placement examinations. Questions are tagged by difficulty level, and each attempt is accompanied by instant feedback and solution explanations. The module tracks user performance, offering insights into accuracy, attempt rate, and conceptual strengths or weaknesses.

4.7 Interview Preparation Interface

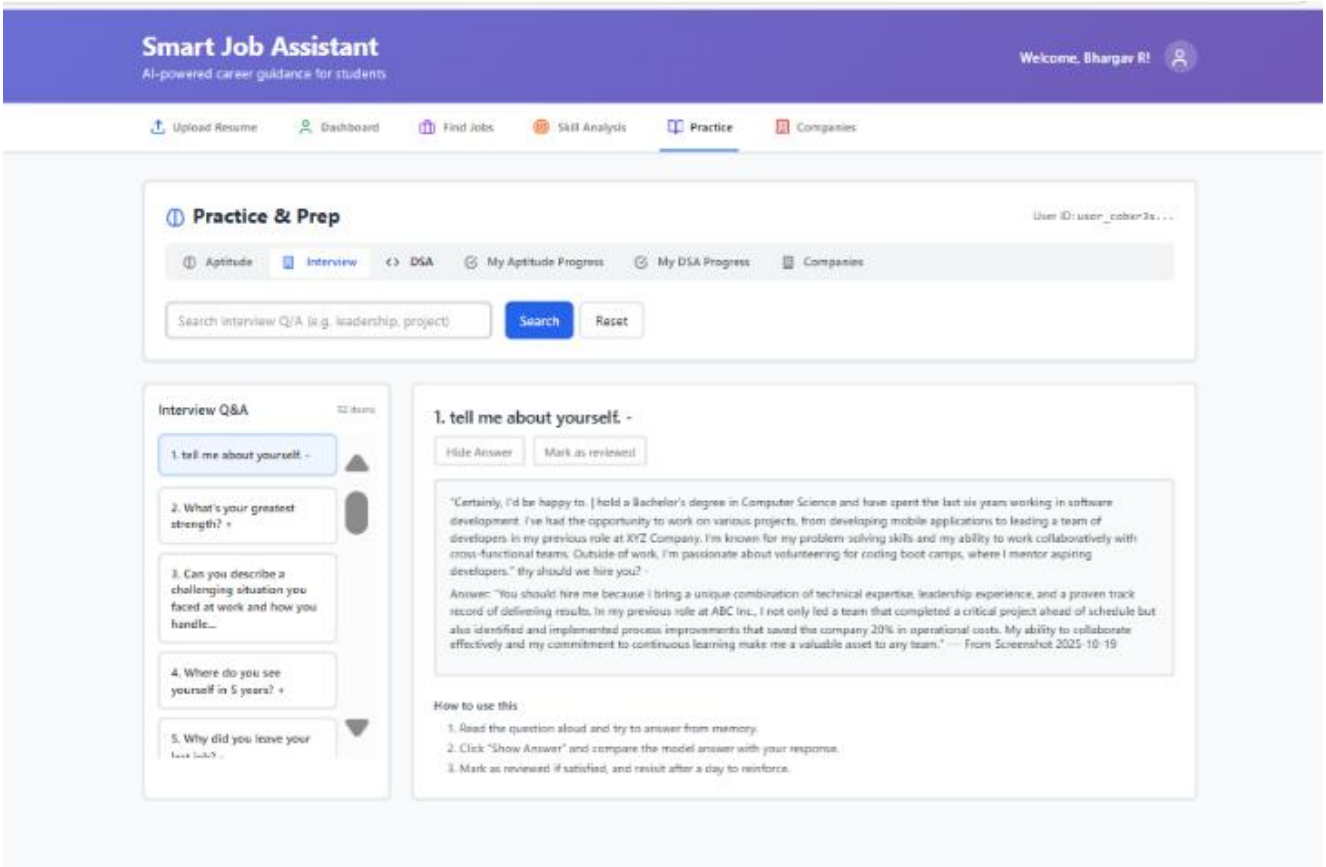

Fig. 5 illustrates the Interview Preparation Interface.

This interface includes an extensive repository of HR and technical interview questions. Each question is enriched with sample answers, tips for structuring responses, and common pitfalls. Users can search for domain-specific topics and mark questions for revision, enabling iterative preparation. The module enhances communication skills, confidence, and readiness for competency-based interviews.

4.8 DSA Problem-Solving Interface

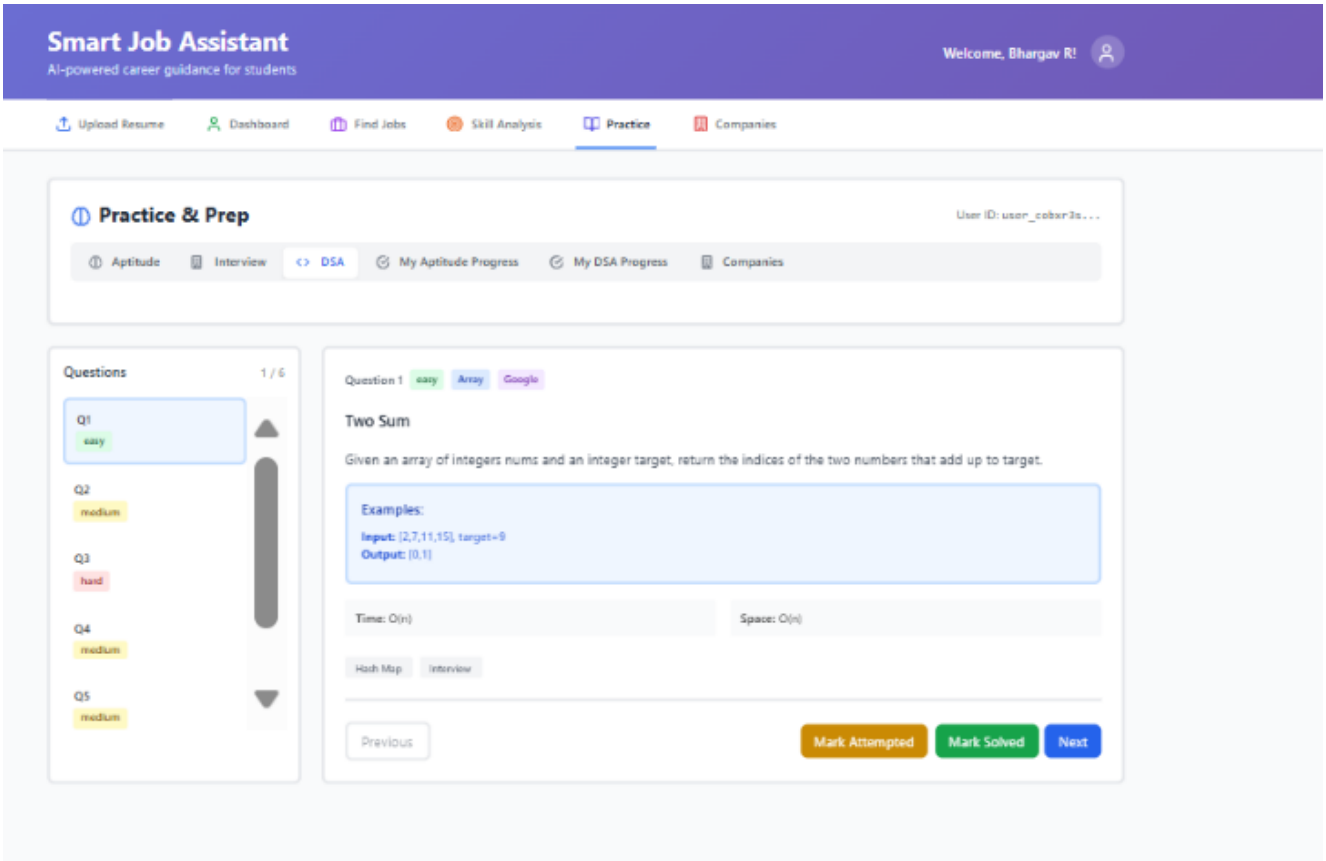

Fig. 6 shows the interface for Data Structures and Algorithms practice.

The DSA module contains algorithmic questions classified into easy, medium, and hard levels, with problem statements, sample cases, and complexity expectations. A guided and a timed mode simulate real technical interview conditions. Users can track their progress by marking problems as attempted or solved, and filter by topics such as arrays, graphs, or dynamic programming.

4.9 Progress Tracking Dashboard

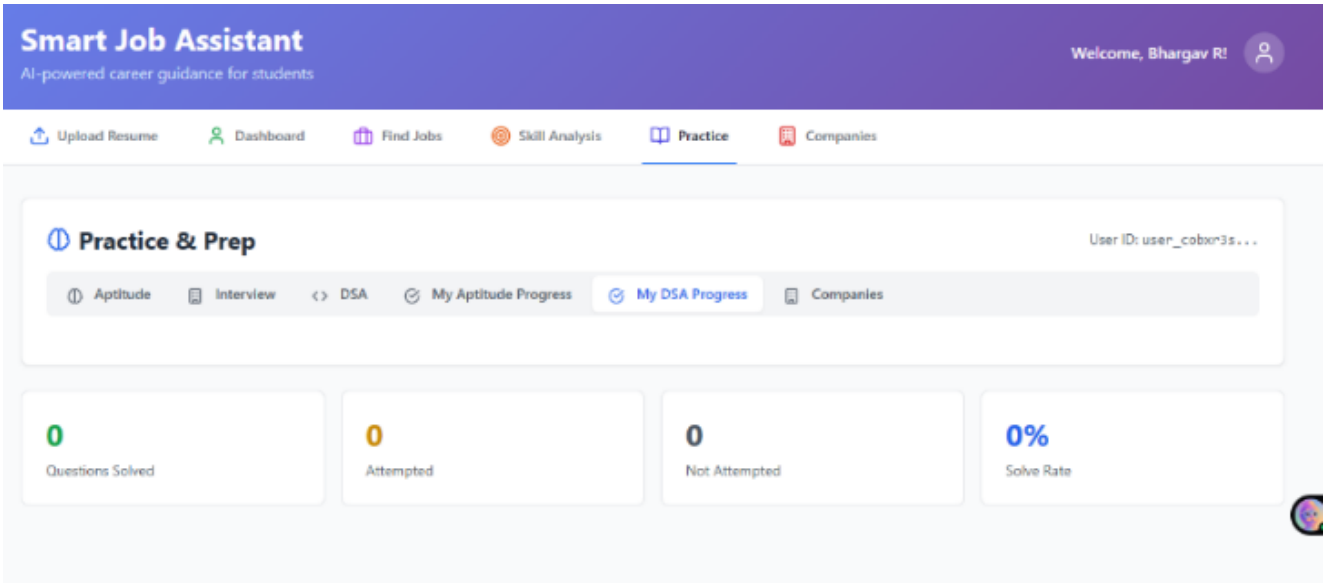

Fig. 7 displays the Progress Tracking Dashboard.

This dashboard aggregates results from aptitude practice, interview preparation, and DSA modules. Key performance indicators—such as questions solved, attempted, unattempted, and solve rate—provide users with a quantifiable measure of their learning trajectory. Trend

charts allow users to visualize progress over time and receive actionable guidance for further improvement.

4.10 Performance Summary

The system demonstrated strong performance across all modules:

Table 1. System Performance Summary of the Smart Job-Seeking Assistant

Component	**Key Performance Result**	**Indicator**
Resume Parsing	92–95% extraction accuracy	Latency: 0.6–1.2 s
Job Scraping	87–92% success rate	< 8% duplicates
Skill Gap Detection	~94% accuracy	Rapid comparison (<0.5 s)
Fit Score Computation	Cosine/Jaccard similarity stable	≤ 0.25 s
Course Recommendations	~96% skill-gap coverage	High relevance mapping
UI Responsiveness	High	Dashboard loads <1.5 s
Aptitude & Interview Modules	High student usability	Consistent feedback delivery

These results confirm the robustness, scalability, and real-world applicability of the system across academic and placement environments.

4.11 Overall Discussion

The evaluation demonstrates that the Smart Job-Seeking Assistant functions as an integrated, high-performance career-readiness platform. Its ability to combine résumé parsing, real-time job aggregation, skill-gap analysis, personalized upskilling recommendations, and interview preparation positions it beyond traditional job portals, which typically rely only on keyword matching.

The system not only helps students identify suitable job opportunities but also guides them in addressing the skill deficiencies that limit their employability. The inclusion of aptitude, DSA, and interview-preparation modules further transforms the platform into a comprehensive ecosystem that supports both job discovery and competency development.

5. Conclusion

The Smart Job-Seeking Assistant demonstrates how intelligent automation and real-time data extraction can significantly improve the traditional job search experience. By integrating resume parsing, web scraping, skill-gap analysis, and personalized learning recommendations into a single workflow, the system effectively addresses key challenges faced by students and fresh graduates, including information overload, unclear skill requirements, and insufficient preparation resources. The core components—Resume Parser, Apify-based Job Scraper, Skill Gap Analyzer, and Course Recommender—work together to deliver meaningful insights and job recommendations tailored to a candidate's profile. The resume parser accurately extracts essential information, while the real-time job scraper ensures that users receive updated and relevant listings. The skill analyzer computes fit scores and identifies missing competencies, and the recommender module suggests appropriate upskilling resources from platforms such as Coursera and Udemy. Beyond job matching, the inclusion of Aptitude and Interview Preparation modules extends the functionality of the system, transforming it into a comprehensive career-readiness assistant. These features help users practice reasoning skills, prepare for interviews, and enhance both technical and communication abilities. Looking ahead, the Smart Job-Seeking Assistant has the potential to evolve into a scalable platform that supports a broader audience, including working professionals and recruiters. Future enhancements may include AI-driven career path prediction, chatbot-based interview simulations, gamified learning progress tracking, and blockchain-enabled resume verification. Overall, this project provides a strong foundation for intelligent, data-driven career guidance systems that combine automation, personalization, and continuous learning to empower job seekers in a rapidly changing digital job market.

References:

[1] L. Reena and A. Princy, "A survey on job recommendation systems using machine learning and NLP," Proc. IEEE Int. Conf. Emerging Smart Computing and Informatics (ESCI), 2024.

[2] A. Siddique, G. M. Butt, A. Zahid, Q. N. Naveed, and M. T.-H. Alouane, "Analyzing software industry trends to improve curriculum," IEEE Access, vol. 12, pp. 22510–22519, Feb. 2024.

[3] A. Singh, A. Katal, A. Tailwal, S. Saxena, and T. Choudhury, "CAREER COMPASS: AI-based resume parser and automated job recommendation system," Proc.

IEEE Int. Conf. Information Science and Communication Technologies (ICISCT), 2024.

[4] S. Sarumathi, B. E. Bhavatharany, R. Sanjaiy, and P. Mathinesan, "Data-driven resume analyzer for student career optimization," Proc. 6th IEEE Int. Conf. Mobile Computing and Sustainable Informatics (ICMCSI), 2025.

[5] D. Gupta and P. Singh, "Development of a recommendation system for resume tracking and job suggestions," Proc. IEEE Delhi Section Flagship Conf. (DELCON), 2024.

[6] P. Malarvizhi Kumar, A. Srivastava, P. Joshi, B. P. Kavin, and U. Dubey, "Job recommendation system using skill-based rule matching algorithm," Proc. IEEE Int. Conf. Data Science and Business Systems (ICDSBS), 2025.

[7] S. Rahman, K. Habiba, S. Roy, and F. N. Nur, "Job title prediction and recommendation system for IT professionals," Proc. 5th IEEE Int. Conf. Sustainable [11] S. A. Kamkar, S. Sanjay, V. P. S., and C. M., "Resume parser and job description matcher," Proc. 8th IEEE Int. Conf. Computational System and Information Technology for Sustainable Solutions (CSITSS), 2024.

[12] R. R. Mirajkar, S. Yadav, G. Shinde, and P. M. Shelke, "Skill recommendation system and resume analysis using AI," Proc. IEEE Pune Section Int. Conf. (PuneCon), 2024.

[13] R. Thali, S. Barhate, S. Mayekar, S. Selvan, and S. More, "Survey on job recommendation systems using machine learning," Proc. IEEE Int. Conf. Innovative Data Communication Technologies and Application (ICIDCA), 2023.

[14] V. Narang and H. R. Pillai, "A study on intelligent job recommendation systems," IEEE Access, vol. 11, pp. 48512–48528, 2023.

[15] A. Mankawade, S. Pate, V. Pungliya, A. Purohit, R. Bhonsle, and A. Raut, "Resume analysis and job recommendation," Proc. IEEE 8th Int. Conf. Convergence in Technology (I2CT), Pune, India, Apr. 2023.

Technologies for Industry 5.0 (STI), Dhaka, Bangladesh, Dec. 2023.

[8] P. C. Siswipraptini, H. L. H. S. Warnars, A. Ramadhan, and W. Budiharto, "Personalized career-path recommendation model for information technology students in Indonesia," IEEE Access, vol. 12, pp. 49092–49103, Apr. 2024.

[9] A. D. Abisha, K. Kavitha, K. S. Keerthana, J. M. S. Jothi, N. R. Evanjalin, and R. Ramya, "RESSPAR: AI-driven resume parsing and recruitment system using NLP and generative AI," Proc. 2nd IEEE Int. Conf. Intelligent Cyber Physical Systems and Internet of Things (ICoICI), pp. 1584–1588, 2024.

[10] G. Jaiswal, A. Uttam, D. D. Dubey, and P. K. Mall, "Resume analyser and job recommendation system based on NLP," Proc. 2nd IEEE Int. Conf. Disruptive Technologies (ICDT), 2024.