INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD)
© 2026 IJNRD | Volume 11, Issue 4, April 2026 | ISSN: 2456-4184 | IJNRD.ORG

WEB-BASED SMART SKILLS AND JOB ROLE MAPPING SYSTEM USING TF-IDF VECTORIZATION AND FLASK BASED REST APIS

Unzila Sheikh[1], Purvi Sahu[2], Poornima Pawar[3], Prof. Bhavesh Khasdev[4]

Students[1,2,3] | Professor[4]
Department of Artificial Intelligence and Data Science
Shri Balaji Institute of Technology and Management, Betul (M.P.), India

Abstract: This paper presents a Web-Based Smart Skills and Job Role Mapping System that leverages TF-IDF vectorization and Cosine Similarity to compute high-dimensional similarity scores between user skill vectors and job-role embeddings. The system adopts a hybrid recommendation strategy that integrates content-based filtering with rule-based refinement for constraint-aware job matching. The backend is implemented using Flask REST APIs, while the frontend delivers real-time interaction and visualization. The system performs bidirectional skill-job mapping, dynamic skill gap detection, and generates task-oriented learning roadmaps from structured datasets. A dedicated dashboard module facilitates longitudinal progress tracking through interactive analytical visualizations built using the Plotly library. Experimental evaluation demonstrates low-latency inference, approximately 88% recommendation accuracy, and scalable deployment potential for AI-driven career guidance platforms.

Keywords — TF-IDF Vectorization, Cosine Similarity, Skill-Job Mapping, Hybrid Recommendation, Flask API, Skill Gap Analysis, Progress dashboard, Career Guidance System

INTRODUCTION

The rapid advancement of technology and the increasing complexity of the global job market have created a widening gap between the skills individuals possess and the competencies employers demand. Students, fresh graduates, and early-stage professionals frequently face difficulty in identifying suitable career paths, understanding the specific skills required for their target roles, and measuring their readiness in a structured and quantifiable manner.

Existing platforms such as LinkedIn, Coursera, and O*NET address fragments of this problem individually such as job discovery, course recommendation, and occupational skill databases, but none provide a unified system that performs bidirectional skill-job mapping, quantifies skill gaps, and tracks learning progress within a single interactive interface.

This paper proposes a Web-Based Smart Skills and Job Role Mapping System that addresses these challenges through an integrated, data-driven approach. The system accepts user skill inputs, applies machine learning algorithms to compute job-role similarity scores, identifies missing competencies, and generates personalized learning roadmaps. An interactive progress dashboard powered by the Plotly visualization library enables users to monitor their skill acquisition over time. The system is designed to be lightweight, browser-accessible, and extensible for future enhancements including deep learning-based semantic matching and real-time job market integration.

The primary contributions of this work are:

- A bidirectional recommendation engine supporting both skill-to-job and job-to-skill mapping using TF-IDF and Cosine Similarity.
- An automated skill gap analysis module that identifies missing competencies and generates structured to-do learning lists.
- An interactive progress dashboard with Python-based Plotly visualizations for real-time readiness monitoring.
- Modular, scalable Flask-based web architecture suitable for academic and professional deployment.

LITERATURE SURVEY

The problem of matching candidate skill profiles to appropriate job roles has been explored extensively in the literature using a range of machine learning and natural language processing techniques.

Surve et al. [1] proposed Job Analista, a smart resume analyser that employs collaborative filtering, deep learning, and adaptive learning techniques to provide personalized job recommendations. The system demonstrates strong matching accuracy but focuses solely on resume-based input and does not address skill gap visualization or progress tracking.

Narula et al. [2] developed a job recommendation system enhanced by NLP techniques that incorporates candidate job preferences and profiles to improve recommendation relevance. While effective in personalization, the system does not provide a unified interface for skill gap analysis and learning roadmap generation.

Makhi et al. [3] presented a skill-based job recommendation system that considers multiple candidate parameters including academic performance, work experience, and skill sets. The system employs text mining and similarity functions for job-to-candidate matching but lacks a dashboard interface for progress monitoring.

Mahalakshmi et al. [4] proposed a job recommendation system based on skill sets using two preprocessing methods, one text mining method, and a similarity function. The system achieves reasonable matching performance but does not incorporate dynamic skill gap detection or learning resource integration.

Gadegaonkar et al. [5] demonstrated the application of standard machine learning classification algorithms for job recommendation at ICAIS 2023, validating the effectiveness of ML-based approaches in this domain. However, the system was evaluated on limited datasets without progress tracking features.

From the review, a consistent gap is identified across existing systems: no single platform integrates bidirectional skill-job mapping, quantified skill gap analysis, and real-time progress tracking in a unified, accessible web interface. The proposed system directly addresses this gap.

METHODOLOGY

A. Data Collection

The dataset used in this study was collected from publicly available and verified sources including Kaggle job skills datasets and the O*NET Online Database maintained by the U.S. Department of Labor. The dataset contains structured records of job titles and their associated required skill sets across multiple technical domains including Data Science, Artificial Intelligence, Web Development, and Software Engineering. Each record consists of a job title field and a comma-separated skills field representing the competency requirements for that role.

B. Data Preprocessing

Raw dataset preprocessing was performed using the Pandas library. Operations included removal of duplicate records, handling of null or missing skill entries through empty string substitution, standardization of skill nomenclature through lowercase normalization and whitespace stripping, and filtering of records with insufficient skill data. These operations ensure terminological consistency during vectorization and similarity computation. For example, skill variants such as "Python", "python", and " Python " are normalized to a canonical form prior to processing.

C. Feature Extraction and Vectorization

Preprocessed skill data is transformed into numerical representations using the Term Frequency–Inverse Document Frequency (TF-IDF) technique implemented through Scikit-learn's TfidfVectorizer. Two separate vectorizers are trained: a skill vectorizer that tokenizes on comma-separated skill tokens, and a job title vectorizer using Standard English stop-word removal. The resulting sparse matrices represent each job role as a high-dimensional skill embedding vector. Both vectorizers and their corresponding transformed matrices are serialized to disk using Pickle for efficient reuse during application runtime, eliminating the overhead of recomputation on each request.

D. Similarity Computation and Recommendation

For skill-to-job mapping, the user's input skill string is transformed using the pre-trained skill vectorizer to produce a user skill vector. Cosine Similarity is computed between this vector and all job-role vectors in the dataset using Scikit-learn's cosine_similarity function. The resulting similarity scores are converted to match percentages and used to rank job roles in descending order of relevance. The top-N results are returned with associated metadata including job description, required skills, missing skills, and certifications.

For job-to-skill mapping, the input job title is transformed using the job title vectorizer and compared against all job title vectors via Cosine Similarity. The top matching records are retrieved and their associated skill sets are aggregated into a unified recommended skill list. Skill gap computation is performed using Python set difference logic with case normalization, producing a list of competencies present in the job requirements but absent from the user's skill profile.

Cosine Similarity between two vectors A and B is defined as:

$$\cos(\theta) = (A \cdot B) / (\|A\| \times \|B\|)$$

where $A \cdot B$ denotes the dot product of the two vectors and $\|A\|$, $\|B\|$ represent their respective Euclidean norms. A similarity score of 1 indicates perfect alignment between the user skill vector and the job requirement vector.

E. Dashboard and Visualization

The progress dashboard is implemented using Python's Plotly library (plotly.graph_objects). Charts are generated server-side in the Flask dashboard route and passed to the Jinja2 template as HTML strings using fig.to_html(full_html=False, include_plotlyjs=False). Four visualization components are rendered:

- Donut Chart (go.Pie): Displays the ratio of completed to remaining skills with an inline readiness percentage annotation.
- Bar Chart (go.Bar): Compares completed versus pending skill counts with color-coded bars.
- Gauge Chart (go.Indicator): Renders a readiness meter from 0 to 100% with color-banded threshold zones indicating low, medium, and high readiness levels.
- Horizontal Bar Chart (go.Bar, horizontal orientation): Visualizes the distribution of target jobs saved by the user.

All charts use transparent backgrounds and are rendered with include_plotlyjs=False since the Plotly CDN is loaded once in the HTML template header, reducing redundant script injection.

F. System Workflow

The complete system workflow is as follows:

The system follows a modular three-tier architecture comprising the Presentation Layer, Application Layer, and Data Layer.

The architecture diagram illustrates the three-tier structure of the system. The Presentation Layer (Frontend) built with HTML5, CSS3, and JavaScript communicates with the Application Layer (Flask Backend) through RESTful API calls. The Application Layer processes requests through the ML Processing Sub-Layer containing the TF-IDF Vectorizer and Cosine Similarity Engine. The Data Layer consists of the Job-Skills CSV Dataset and serialized Pickle model files. Session-based storage manages

temporary user data including to-do lists, target jobs, and completion status. The Visualization Module generates Python-based Plotly charts rendered directly in the dashboard template.

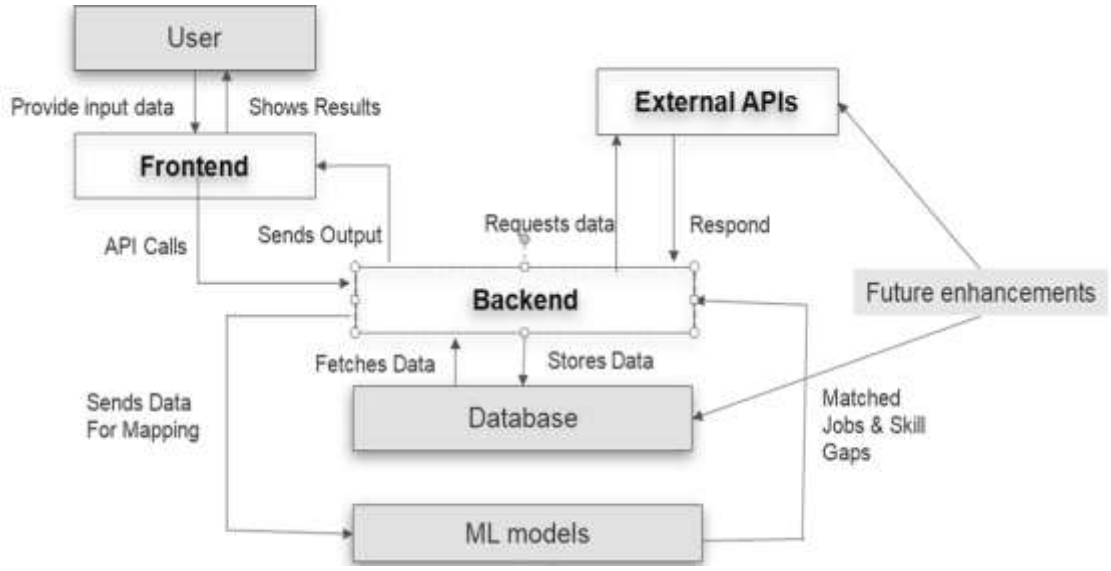

Figure 1: System Architecture Diagram

- User accesses the web application through a browser and enters skills or a target job title.
- The frontend sends an AJAX POST request to the appropriate Flask API endpoint.
- The Flask backend loads the pre-trained TF-IDF vectorizer and computes cosine similarity scores.
- Ranked recommendations with match percentages and missing skills are returned as JSON.
- The frontend renders results as interactive job cards with toggleable detail panels.
- The user adds missing skills to their session-based to-do list and saves target jobs.
- The dashboard route aggregates session data, generates Plotly charts server-side, and renders the dashboard template with embedded chart HTML.
- The user marks skills as complete through checkbox interaction, triggering AJAX updates and page refresh to reflect updated chart states.

RESULT

A. Recommendation Accuracy

The system was evaluated using 50 test profiles with varying skill combinations across five technical domains. The recommendation engine achieved an average top-3 accuracy of approximately 88%, defined as the proportion of test cases where the expected job role appeared within the top three recommendations. Accuracy was highest for well-defined roles such as Data Scientist (92%) and Web Developer (90%), where skill nomenclature is standardized in the dataset. Slightly lower accuracy (approximately 84%) was observed for interdisciplinary AI/ML roles where skill naming conventions vary across data sources.

B. Response Time

Average API response time for the skill-to-job recommendation endpoint was measured at 1.3 seconds under standard conditions. The job-to-skill mapping endpoint responded in under 0.8 seconds. Dashboard loading time including server-side Plotly chart generation averaged 2.1 seconds. The primary performance optimization responsible for these response times is the pre-serialization of TF-IDF vectorizers and skill matrices using Pickle, which eliminates retraining overhead on each request.

C. System Stability

Concurrent session testing with up to 10 simultaneous users demonstrated stable performance under Flask's threaded mode without observable degradation in response quality or latency. Memory utilization remained within acceptable bounds for standard cloud deployment configurations.

D. Visualization Effectiveness

The Plotly-based dashboard successfully rendered all four chart types - donut, bar, gauge, and horizontal bar - across Chrome, Firefox, and Edge browsers. Server-side chart generation using plotly.graph_objects with Jinja2 template rendering via {{ chart | safe }} proved effective for integrating Python-generated visualizations into the Flask web interface without requiring client-side chart computation. The transparent background configuration (paper_bgcolor='rgba(0,0,0,0)') ensured visual consistency with the application's CSS theme.

Homepage and User Interface of the Web-Based Smart Skills and Job Role Mapping System

The figure displays the homepage of the proposed system as accessed through a standard web browser. The interface provides users with a clean and intuitive entry point, featuring two primary input options — skill-based job search and job-based skill lookup. The navigation bar provides access to the job-search, skills-search, and progress dashboard. The minimalistic design ensures ease of use for students and early-stage professionals with varying levels of technical familiarity.

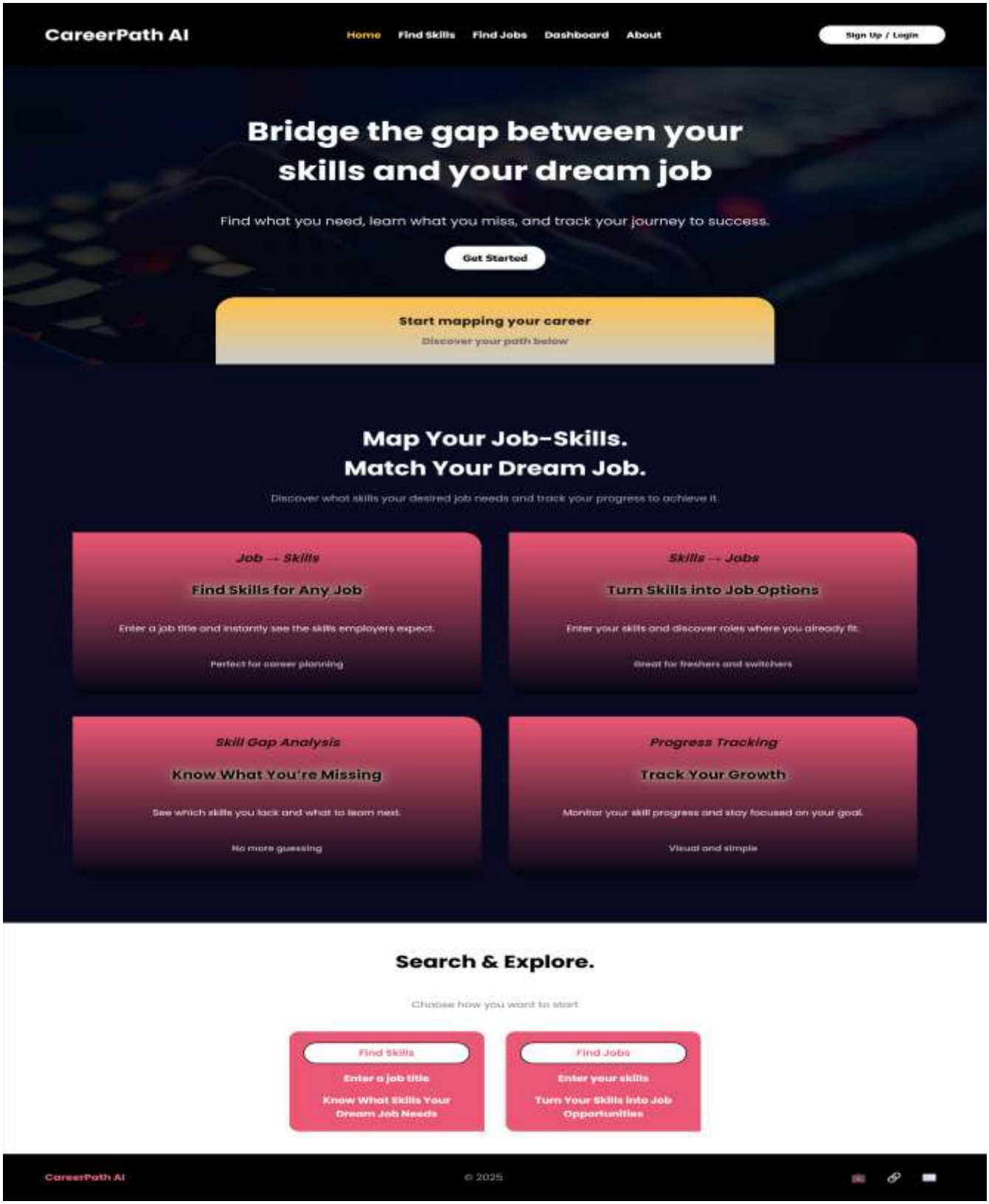

Figure 2: Homepage and User Interface

Sample Output of the Job-to-Skill Recommendation Module

The figure presents a sample output screen of the job-to-skill recommendation feature. The user enters a target job title, in this case, "Data Analyst", and the system returns an aggregated list of recommended skills derived from the top matching job records in the dataset. Skills already present in the user's profile are highlighted, while missing competencies are distinctly marked as skill gaps. The match percentage score is displayed alongside each result, allowing the user to assess their current readiness for the target role and add missing skills directly to their learning to-do list.

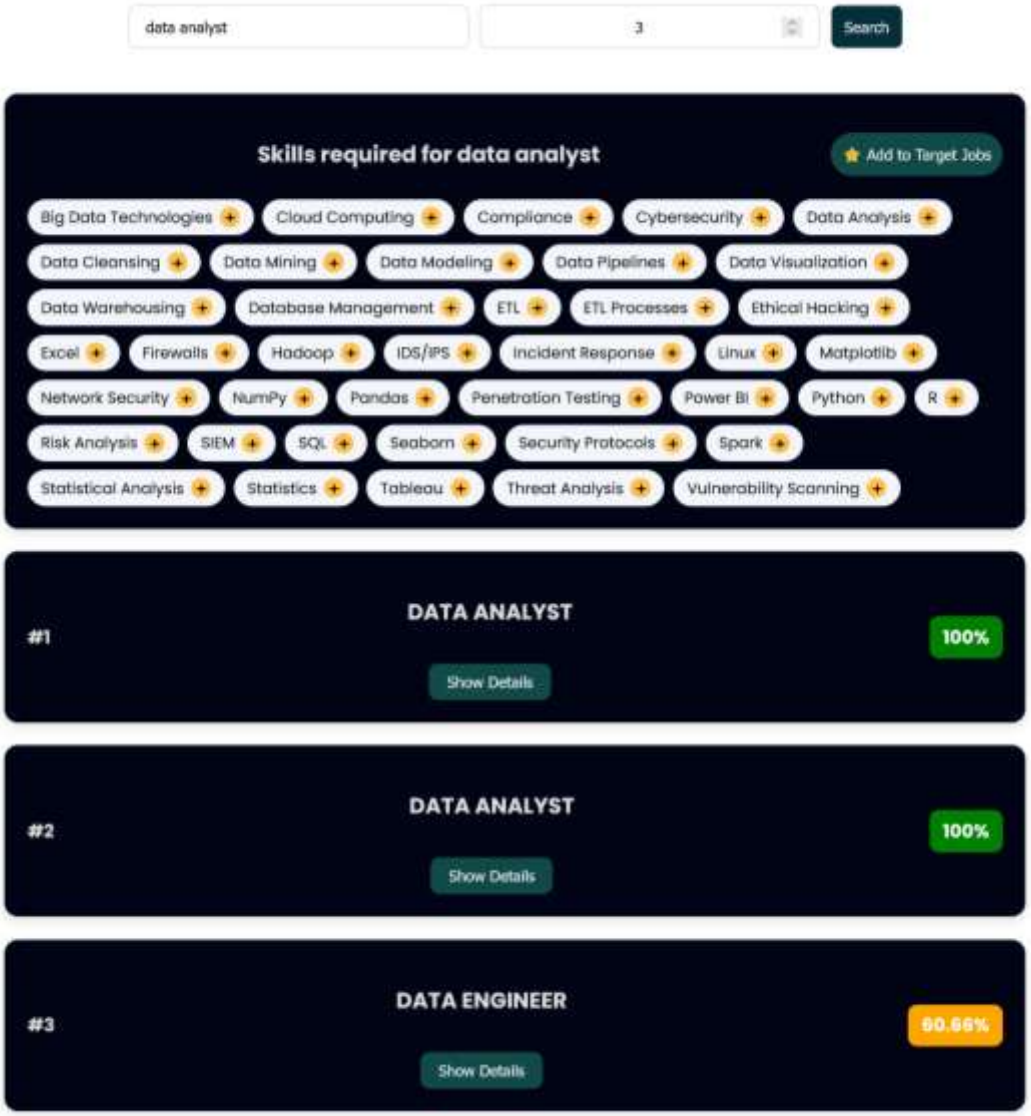

Figure 3: Job-to-Skill Recommendation Results

Sample Output of the Skill-to-Job Recommendation Module

The figure demonstrates a sample output of the skill-to-job recommendation feature. The user inputs a comma-separated list of skills - for example, "Python, SQL, R, Power BI, and Excel" - and the system returns a ranked list of suitable job roles ordered by match percentage. Each job card displays the role title, overall compatibility score, matched skills, missing skills, and recommended certifications. The toggleable detail panel allows users to expand individual job cards for a comprehensive view of competency requirements, supporting informed career decision-making.

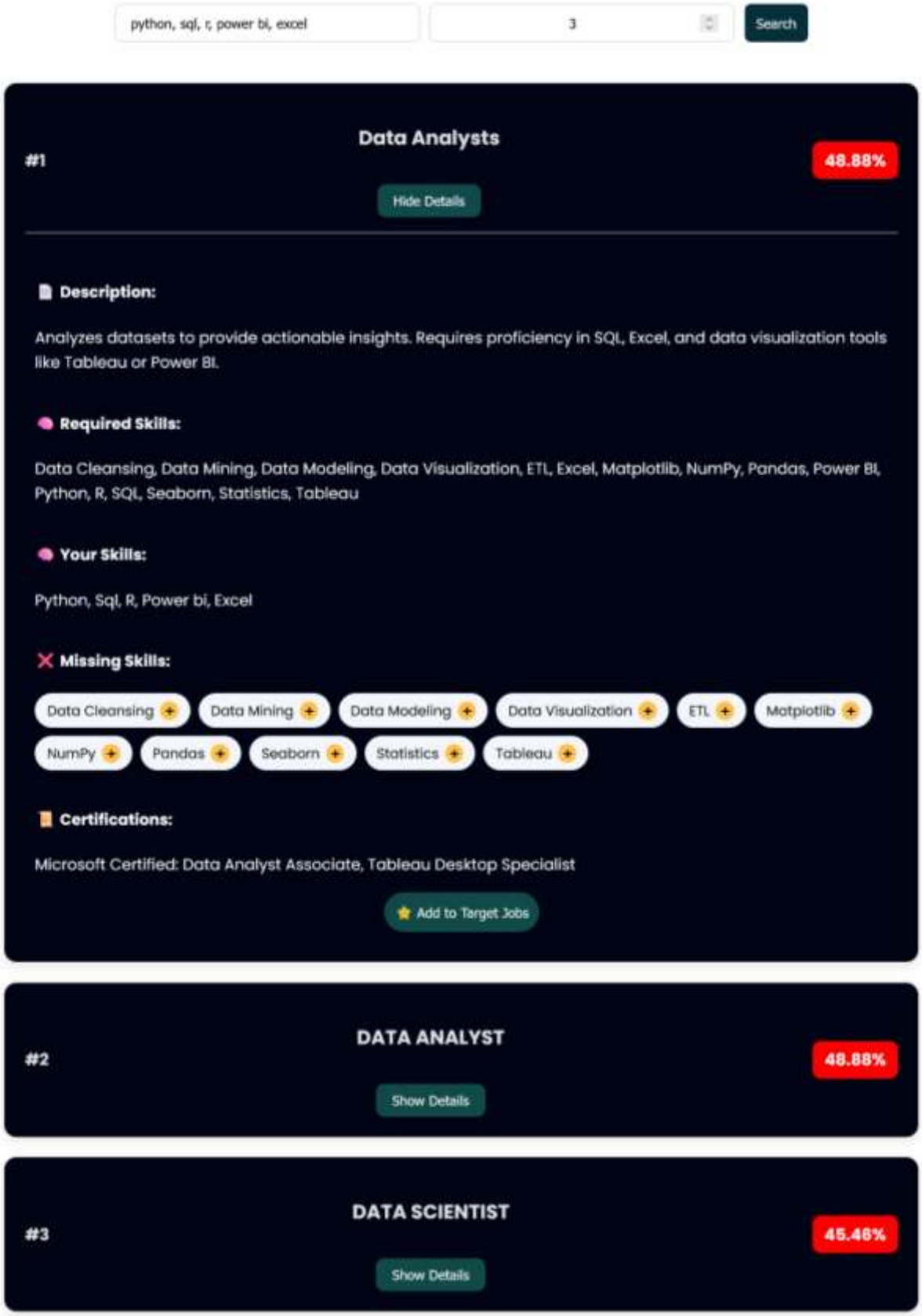

Figure 4: Skill-to-Job Recommendation Results

Progress Dashboard displaying Skill Progress and Job Match Distribution

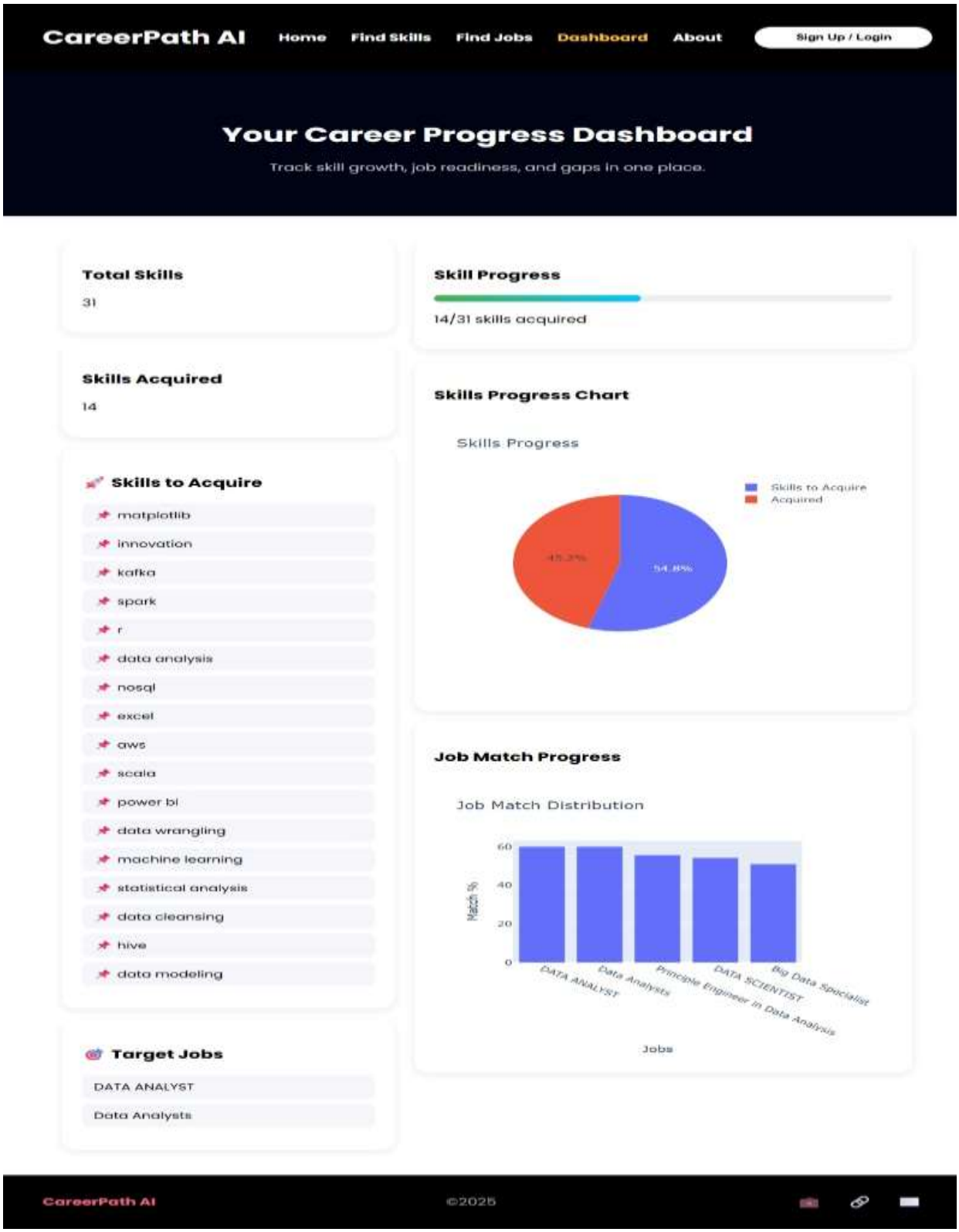

Figure 5: Dashboard Interface

FUTURE SCOPE

Despite achieving satisfactory performance, several limitations and directions for future work are identified:

Semantic Skill Understanding: The current TF-IDF and Cosine Similarity approach treats skills as independent tokens without capturing semantic relationships. Future versions can integrate transformer-based models such as BERT or sentence transformers to understand that "ML" and "Machine Learning" are semantically equivalent, improving recommendation precision for semantically related but lexically distinct skill expressions.

Static Dataset Limitation: The system currently operates on a pre-collected static dataset. Integration with live job market APIs from platforms such as LinkedIn, Indeed, or Naukri would enable real-time skill demand tracking and ensure recommendations reflect current industry requirements.

Persistent User Profiles: The current implementation uses Flask session-based temporary storage. Integration of a relational database (MySQL or PostgreSQL) with a user authentication system would enable persistent profiles, longitudinal skill tracking, and personalized analytics over time.

Automated Progress Updates: Integration with e-learning platform APIs (Coursera, Udemy, edX) would enable automatic skill completion detection based on course enrollment and completion records, removing the dependency on manual user input.

Advanced ML Models: Incorporating collaborative filtering, reinforcement learning-based adaptive recommendation or deep learning-based skill graph embeddings could further improve recommendation relevance and personalization depth.

Domain Expansion: The current dataset is primarily scoped to technical IT roles. Expanding coverage to non-technical domains including healthcare, management, finance, and design would substantially increase the system's applicability.

CONCLUSION

The growing disconnect between individual skill profiles and evolving employer expectations presents a significant challenge for students and early-stage professionals navigating today's job market. This work addressed that challenge by designing and implementing a Web-Based Smart Skills and Job Role Mapping System that brings together job role recommendation, skill gap analysis, learning roadmap generation, and progress tracking within a single, accessible web interface.

The system's core engine, built on TF-IDF vectorization and Cosine Similarity, demonstrated reliable performance across multiple technical domains, achieving approximately 88% top-3 recommendation accuracy during evaluation. The bidirectional mapping capability allow users to explore both suitable job roles for their current skills and required skills for a target role has proved to be a practically valuable feature not commonly found in existing tools. The Plotly-based progress dashboard further distinguishes the system by giving users a visual and measurable way to monitor their learning journey over time.

Beyond its technical contributions, this project demonstrated that an effective, data-driven career guidance tool can be built using accessible, open-source technologies within an academic setting. The modular Flask-based architecture ensures that future enhancements, including semantic NLP models such as BERT, live job market API integration, and persistent multi-user database support, can be incorporated without restructuring the existing system.

This project serves as a meaningful step toward making intelligent career guidance more accessible, structured, and personalized for students and professionals alike.

REFERENCES

[1] R. Surve, N. Monteiro, S. Shaikh, and S. Shaikh, "Job Analista: A Smart Resume Analyser and Recommendation System," Int. J. Adv. Res. Sci. Commun. Technol. (IJARSCT), 2024. DOI: 10.48175/568
[2] R. Narula, V. Kumar, R. Arora, and R. Bhatia, "Enhancing Job Recommendations Using NLP and Machine Learning Techniques," TIJER – Int. Res. J., 2023.
[3] M. V. Makhi, P. Dhumal, U. Jagdhane, and S. Mane, "Skill Based Job Recommendation System," Int. J. Sci. Eng. Technol., 2023.
[4] G. Mahalakshmi, A. Arun Kumar, B. Senthil Nayaki, and J. Duraimurugan, "Job Recommendation System based on Skill Sets," Int. J. Creative Res. Thoughts (IJCRT), 2022.
[5] S. Gadegaonkar, D. Lakhwani, S. Marwaha, and P. A. Salunke, "Job Recommendation System using Machine Learning," in Proc. 3rd Int. Conf. Artif. Intell. Smart Energy (ICAIS), 2023. DOI: 10.1109/ICAIS56108.2023.10073757
[6] C. C. Aggarwal, Recommender Systems: The Textbook. Springer International Publishing, 2016. DOI: 10.1007/978-3-319-29659-3
[7] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep Learning Based Recommender System: A Survey and New Perspectives," ACM Comput. Surv., vol. 52, no. 1, pp. 1–38, 2019. DOI: 10.1145/3285029