AI-Based Internship Recommendation Engine for PM Internship Scheme

Ms. Rama Lakshmi[1], P. Ajay[2], K. Arjun[3], B. Nischay[4]
[1]*Assistant Professor, Dept. of Computer Science and Engineering, Matrusri Engineering College*
[2,3,4] *UG Scholars, Dept. of Computer Science and Engineering, Matrusri Engineering College*

***Abstract*—Students who want assistance in finding proper internship positions face challenges when they attempt to use large internship platforms which lack personalized support to match their abilities with suitable job vacancies. The Internship Recommendation Engine functions as an artificial intelligence system which matches skills with specific internship requirements based on candidate preferences for field of study and geographic location and their level of experience. The system employs Sentence Transformer embeddings to comprehend the meaning of both user profiles and internship requirements instead of depending on keyword matching. The system uses FAISS indexing technology to store representations which enables users to conduct rapid and precise similarity searches across a complete database of actual internship job postings. The system uses a recommendation ranking formula which evaluates multiple factors including semantic similarity and skill fit and location preference and experience level to establish priority order. The general filters which users select activate the domain diversity mechanism to create new entries which stop the system from producing identical results. The resume upload process enables automatic skill extraction for technical skills through natural language processing technology. The system uses a large language model to identify skill gaps for each role recommendation and it develops structured learning pathways. The complete system operates as a web application which provides five top internship recommendations together with detailed information about required qualifications. The evaluation results demonstrate that this method outperforms basic keyword filtering by accurately matching candidates with suitable job positions.**

***Index Terms*—Internship Recommendation, Semantic Similarity, Sentence Transformers, FAISS, Skill Gap Analysis, Natural Language Processing.**

I. INTRODUCTION

Students who have just started their professional development face difficulties in finding their perfect internship position. The PM Internship Scheme establishes specific requirements which need to be met by candidates. The program provides students with real-world experience yet they must undertake a separate difficult task which involves using extensive platforms to locate job opportunities that match their professional experience and skills and geographic area. The majority of eligible students come from rural and semi-urban regions which lack career counseling services that would help them understand how to match their abilities with job requirements.. Because of this, a lot of students either apply at random without knowing if the position is a good fit for them or miss out on the opportunity entirely.

Traditional internship platforms such as Internshala, LinkedIn, and Naukri use keyword-based search systems which require students to input their exact search terms to find matching job listings. The system operates with inherent restrictions which limit its performance. A student who understands PyTorch will not qualify for deep learning internship positions because the job posting uses the term TensorFlow instead of PyTorch.

The researchers developed an AI-based Internship Recommendation Engine to solve the problem which helps students who meet PM Internship Scheme requirements and have low digital literacy skills. The method assesses a student's abilities to measure their compatibility with actual internship requirements which traditional systems achieve through basic keyword matching. The system uses Sentence Transformer embeddings to identify deeper relationships between words instead of relying on

superficial word matching. For rapid and precise retrieval across hundreds of listings, it also employs FAISS vector search. The results are then arranged using a weighted ranking system that combines experience level, geography, desired domain, and skill similarity.

The system uses natural language processing to extract technical skills from student resumes which it processes to identify skill gaps for each recommended position and it utilizes a language model to produce a structured educational path. A mobile-friendly web application delivers five internship recommendations which it presents in order of their importance.

The main goals of this project include building an internship recommendation system which uses Sentence Transformer embeddings and FAISS vector search to achieve better performance than regular keyword-based filtering methods; developing a four-part weighted ranking system which evaluates skill match and semantic similarity and location preference and experience level all at once; creating a soft domain diversity mechanism which stops any single domain from controlling the results of open-ended searches; establishing a resume analysis module which extracts technical skills from PDF documents that users upload; using the Groq Llama 3 language model to create personalized skill gap insights and learning roadmaps through AI-based technology; and creating a production-ready complete system which functions as a full-stack web application that users can access through any browser without the need for installation.

II. LITERATURE SURVEY

Intelligent recommendation systems, skill-based retrieval, and semantic matching have all drawn a lot of interest lately. This section reviews the most relevant prior research and shows how the proposed system improves and builds upon it.

Koren, Bell, and Volinsky have demonstrated that matrix factorization techniques regularly outperform neighborhood-based collaborative solutions at scale by decomposing user-item interaction matrices into latent factor representations. Collaborative filtering is powerful, but it has a well-known cold-start problem that keeps it from recommending things to users who have never used it. Since students using this system for the first time lack historical data, pure collaborative filtering is not relevant in this domain, which promotes a content-based approach.

Reimers and Gurevych [Reimers and Gurevych2019a] proposed Sentence-BERT (SBERT), which applies a Siamese network architecture to fine-tune BERT on sentence pair similarity datasets. When the related sentences have high cosine similarity scores, the model generates semantically meaningful sentence embeddings. This work directly supports the choice of the all-MiniLM-L6-v2 model in the proposed system, which is a distilled derivative of SBERT trained on more than one billion sentence pairs, while retaining over 99% of the parent model's performance at a quarter of the computational cost.

Devlin and his colleagues introduced the Bidirectional Encoder Representations from Transformers BERT model which marks a major advancement in the field of natural language processing. BERT creates word embeddings that provide context-based word meanings through its training on masked language modeling which uses extensive language data. BERT differs from previous systems which assigned one static vector to each word because it generates multiple word representations that depend on the word's specific function within a sentence. BERT possesses strong semantic abilities but its design does not support functions that require sentence similarity comparison. BERT fails to match performance of basic semantic similarity evaluations when researchers use simple averaging techniques to create sentence representations from token embeddings. The system requires a dedicated sentence embedding model because BERT pooling results cannot provide accurate and computationally efficient semantic representations needed for internship matching.FAISS (Facebook AI Similarity Search) is an open source library for efficient similarity search and clustering of dense vectors. It was first introduced by Johnson, Douze and Jégou. FAISS provides various types of index structures, from exact search to approximate search accelerated by GPU. The proposed system uses IndexFlatIP to compute accurate inner product similarity. Theoretically this is equivalent to cosine similarity, as all vectors are L2 normalized, and therefore scores are strictly bounded in the range zero to one, with no approximation error.

Zhu et al. propose a person-job fit system based on joint representation learning, in which Bi-LSTM networks are used to encode candidate profiles and job

descriptions into a unified semantic space. This project immediately instantiates the key finding of their study, namely that meaningful comparison is possible by using the same model to include both the user and the object. The skills query of the student and the concatenated text field of the internship are encoded by the same all-MiniLM-L6-v2 model.

Qin et al. [83] developed a hybrid approach for job recommendation that combines deep semantic matching with signals from structured attributes such as location and experience level. The hybrid approach outperforms either signal alone. This finding directly validates the four-component weighted ranking formula in the proposed system, which combines semantic similarity with structured attribute scores for skill match, location preference, and experience alignment.

Sayfullina, Malmi, and Kannala demonstrated that LSTM-CRF models fine-tuned on skill-annotated job advertisement corpora achieve high precision in technical skill extraction. However, general-purpose NER models trained on newswire or Wikipedia text perform poorly on technical job vocabulary because the training domain is too distant. This motivated the adoption of a curated dictionary-based extraction approach in the proposed system, which prioritizes longer multi-word technical phrases and covers domain-specific tools such as FAISS, FastAPI, and pdfplumber that are absent from standard NER corpora.

Covington, Adams, and Sargin described YouTube's two-phase deep neural network recommendation architecture, comprising a candidate generation stage followed by a ranking stage. This architecture directly parallels the proposed system's FAISS candidate retrieval phase followed by the weighted ranking phase, confirming that two-phase pipelines are effective for large-scale personalized recommendation.

The existing literature demonstrates that no previous research work has created a practical application which unifies all three elements of semantic embedding retrieval, structured attribute ranking, and domain diversity with automated skill extraction from resumes and LLM career guidance. Recommendation systems existing today either use semantic matching for their entire operation or they extract skills but do not provide users with knowledge gaps and learning pathways. The research system combines all major research advancements from various fields to develop a single tool which delivers practical benefits. The system uses all-MiniLM-L6-v2 for semantic encoding according to SBERT literature and the FAISS-based retrieval pipeline gets validation through extensive similarity search studies and the weighted ranking formula receives validation from hybrid recommendation studies and the LLM-powered roadmap generation functions because of progress in instruction-following language models. The system operates as a unified whole which delivers greater functionality than its individual components because it fills a research requirement which no individual study has achieved.

III. PROPOSED METHODOLOGY

The suggested system is constructed as a full-stack web application with Next.js for the frontend and FastAPI for the backend. Each of the six stages that make up the overall technique builds directly on the one before it. Each component is explained in detail in the below subsections.

A. Gathering and Preparing Data

An initial dataset of 659 items was produced by gathering internship data from two portals, Internshala and Open The total number of unique posts reached 642 after the removal of all duplicate entries. Each post contains the following information role title, firm name, domain, required skills, job description, location, work format, experience level, stipend, and application link. The process of data cleaning used organized methods to identify and remove entries that did not meet quality standards. The two scripts scan_ml_mismatch.py and scan_all_dirty.py were used to study four different scenarios which included machine learning skills assigned to web or mobile roles and web skills linked to AI positions and mobile skills linked to unrelated domains and soft skills placed in technical job listings. The manual review process resulted in the removal of approximately 47 records which showed inconsistency among the highlighted entries. The final dataset contained 595 internships which covered 20 different professional fields. The job title, platform source, necessary skills, and the first few lines of the job description were combined to create a combined text block for each listing. The text block serves as input for the embedding model.

B. The Process of Generating Semantic embedding begins

The all-MiniLM-L6-v2 Sentence Transformer model was selected because it provides a precise output while maintaining efficient computational performance. The system transforms text into 384-dimensional numerical vectors which capture all textual elements except for exact word matches between the original text and the new format. The system operates effectively under resource constraints because it presents a streamlined version of BERT which was developed through extensive training on vast sentence pair datasets.

The model processes all 595 internship text blocks to create their corresponding vector representations during the server establishment. The system performs L2 normalization on each vector which enables users to compute cosine similarity through the same method they use for dot product calculations. FAISS provides similarity scores which range between zero and one without any interruptions to the score.

C. Vector Search FAISS

FAISS allows Facebook AI Similarity Search to conduct candidate matching through its rapid search method. The system uses an IndexFlatIP index to determine the precise inner product values between two vector inputs. The procedure functions like cosine similarity because it requires L2 normalization for all vector data. The skills query submitted by a user undergoes normalization before the system uses the embedding model to convert it into a vector format. FAISS retrieves the top 50 internship vectors which are the most similar to the input data. The system retrieves fifty candidates to maintain enough variety during the subsequent stages of ranking and filtering.

D. Domain Filtering in Two Stages

The system applies two filtering processes to the top 50 FAISS search results after users select their desired domain. The first stage filters all entries which match the selected domain for retention. The system applies DOMAIN_GROUPS predefined grouping to extend its search into associated domains when there are less than five hits after the first stage. The machine learning group includes all three categories of data science, data analytics, and artificial intelligence. The system guarantees that even with limited domain requests, it will return enough relevant results.

E. Formula for Weighted Ranking

The system uses four weighted components to calculate the final score of all filtered search results. The final score is computed as follows:

$$\text{final\_score} = 0.40 \times \text{s\_sim} + 0.35 \times \text{s\_match} + 0.15 \times \text{loc} + 0.10 \times \text{exp} \quad (1)$$

The weight of semantic similarity stands at 0.40 because it measures how closely the student profile corresponds with the internship requirement. The role demands measurement shows matching skills between student abilities and role responsibilities. The location preference measurement contributes 0.15 while experience level alignment accounts for 0.10 of the total measurement. The cosine similarity scores from FAISS determine the most important factor because they assess overall meaning alignment.

The system compares the user's skills with the internship requirements after both data sets undergo lowercase conversion to assess skill matching. The scoring system assigns 1.0 for exact matches, 0.9 for remote jobs, 0.5 for locations that are not specified, and 0.2 for mismatches based on location preference. The experience level matching system assigns a score of 0.3 when the user's level is insufficient and 0.8 when experience is not stated and 1.0 for an exact match. The final ranking process combines the four scores according to the established weight system.

F. The Process of Analyzing Resumes along with Assessing Soft Domain Diversity

The system prevents any single domain from dominating the results when users select no specific domain. The system applies a penalty multiplier of 0.90 which increases based on the number of times a domain has already appeared after it completed its assessment of the top 20 possible domains. The results are rearranged to return the top five options. Any result which has a semantic score below 0.05 will be removed from the system.

PDFplumber is used to parse submitted PDF files page by page for resume-based searches. In order to prevent incomplete matches, a skill extraction script prioritizes longer phrases over shorter terms when comparing the text to a curated list of more than 200 technical abilities. The FAISS search pipeline receives identified skills directly. By comparing user skills to job criteria, skill gaps are calculated for each recommended role.

IV. SYSTEM DESIGN

A. System Architecture

The proposed system includes a client-server design which keeps the display and application and intelligence and data levels completely separate from one another. The Next.js frontend development team uses TailwindCSS and Typescript to create a mobile-friendly design which achieves responsive functionality. The backend system provides RESTful API endpoints which use JWT-based authentication to secure access and it operates with FastAPI on Python 3.11. The layers use Axios as their client-side HTTP library to establish HTTP/REST communication between them.

The recommendation system operates through five separate stages which it follows during its execution. The user logs in through JWT authentication during Phase 1 to access the dashboard. The skills query which was submitted by the user gets transformed into a 384-dimensional vector which FAISS uses to find the top 50 candidates. The results of Phase 3 domain filtering begin with exact matching which becomes wider through DOMAIN_GROUPS expansion when there are not enough results. The final five recommendations are produced through Phase 4 which uses weighted ranking and soft domain diversity penalties. The system uses resume upload to enable Phase 5 which starts PDF processing and skill extraction and gap calculation and Groq Llama 3 roadmap creation. The phased design of the system allows complete testing of each stage while enabling improvements which do not impact other system components.

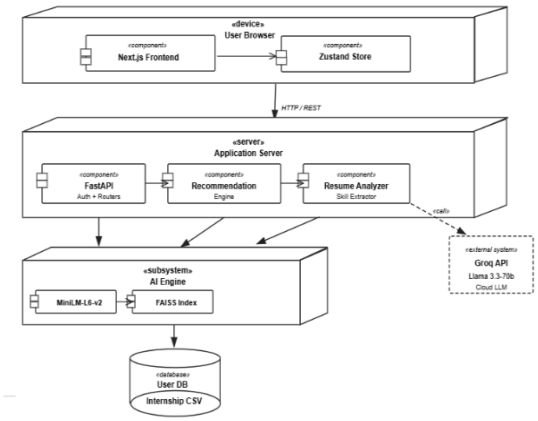

Fig. 1. System Architecture of the Proposed Internship Recommendation Engine

B. Recommendation Pipeline

The recommendation pipeline operates in five stages. Users start the process by submitting their skills along with their optional filters which the system transmits to the backend. The input data gets transformed into a semantic vector through the all-MiniLM-L6-v2 model. FAISS then retrieves 50 internship candidates with similar qualifications to maintain candidate diversity through this process. The system uses domain-based filtering methods to improve its results which include expansion to related domains when necessary. The system uses a weighted ranking method to choose five internships which it displays to users through the interface.

C. Implementation Details

The system uses machine learning models together with backend technologies to deliver recommendations which are both quick and precise. The all-MiniLM-L6-v2 Sentence Transformer produces semantic embeddings from internship data and user inputs which are stored in FAISS (IndexFlatIP) for fast similarity searches. A weighted ranking mechanism improves result accuracy through its evaluation of semantic similarity and skill alignment and additional factors. The system uses FastAPI with Python 3.11 to manage API requests and system operations while the LLaMA 3 model (via Groq API) generates personalized learning roadmaps for users. The implementation creates a recommendation pipeline which can grow and operates with intelligent capabilities.

V. RESULTS AND ANALYSIS

A. Dataset Overview

The final dataset, used for evaluation, includes 595 internship listings from Internshala and Open Internships. The initial merged dataset had 659 entries. After removing 17 identical records, 642 unique listings remained. The data quality assessment process removed 47 records which contained incorrect skill information because the documented skills did not match the internship position and field requirements. The cleaned dataset contains complete data for all 13 columns while it includes data from 10 different geographical regions and 20 distinct professional areas.

B. Evaluation Methodology

The testing process used ten established test cases to evaluate system performance by testing different types of queries. The test cases included domain-specific searches and edge cases which used unrelated talents and single-skill queries and multi-skill domain-specific searches. Evaluation of each test case was based on four primary criteria: the domain accuracy of the returned results, the relevance of the titles to the query, the validity of the apply links, and the precision of the final scores in representing match quality.

C. Recommendation Accuracy

The system provided five pertinent internship recommendations for each of the ten test instances. The range of scores varied according to the specificity of the query. The mobile development study produced the highest overall results because all five tested cases received scores between 0.503 and 0.559 which contained relevant domain matches that included flutter mobile-specific names. The Python query which tested a single skill received its highest score from a perfect skill match rating of 1.0 because the top two results scored 0.727 and 0.706 respectively. The general inquiries which include machine learning without filters received scores between 0.473 and 0.529 while the top three results belonged to the machine learning category.

The score differences between various query types demonstrate expected results from the weighted ranking formula instead of showing any system performance inconsistencies. Single-skill Python and domain-filtered mobile development queries produce higher scores due to the skill match component having a weight of 0.35 which achieves almost complete overlap between user input and internship requirements. The machine learning without filters query produces lower scores because the skill match component distributes across more internship requirements although the semantic similarity component maintains its strong performance. The score variation functions as a valuable signal which shows student profile matching with recommended roles because it helps students evaluate their results based on specific qualifications.

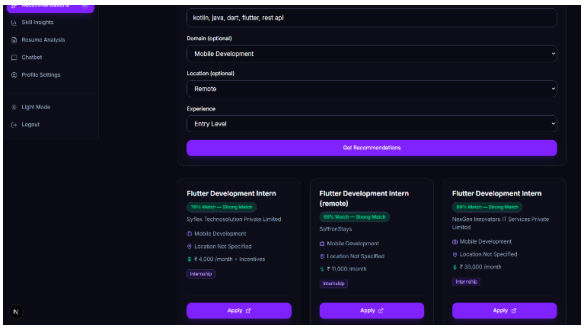

Fig. 2. Internship recommendations generated for a mobile development query

D. Two-Stage Domain Filter Performance

The two-stage domain filter test used the data science domain which contains 12 entries from the dataset as testing material. The output remained useless before the system change because the exact match filter returned only a single result. The system succeeded in retrieving five relevant results which scored between 0.400 and 0.447 after the two-stage process was initiated and the system expanded its search to related fields of data analytics machine learning and data engineering..

E. Edge Case Handling

A skills query that had nothing to do with cooking or baking was used to test the system. For each of the five outcomes, skill match scores were 0.00 and semantic similarity ratings fell to the range of 0.12 to 0.15. The final scores were between 0.20 and 0.22, far below any significant threshold for recommendations. The system completely filtered out all of the results because they were all below the minimal skill match criterion of 0.05, and it showed the message "No recommendations found, try different skills and filters". This attests to the system's elegant handling of irrelevant input, free from mistakes or incorrect suggestions.

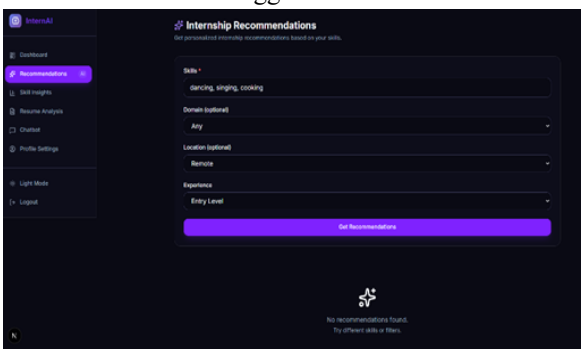

Fig. 3. System response for an unrelated skills query displaying a no recommendations found message.

F. Comparison with Keyword-Based Filtering

Conventional keyword-based search would not be able to produce results that are semantically linked if certain terms were missing from the listing. For example, if the job description just includes TensorFlow, a user who is skilled with PyTorch can miss deep learning internships. The system's semantic embedding approach captures such correlations, as demonstrated by the machine learning and artificial intelligence test cases where adequate results were obtained despite partial or indirect skill overlap. The edge case test further confirms that the system correctly assigns low confidence scores when no semantic relationship exists, in contrast to keyword search, which can produce zero results or irrelevant matches without a confidence indicator.

G. System Response

The recommendation engine loads all 595 internship embeddings into the FAISS IndexFlatIP at server startup to enable fast query retrieval because it needs to retrieve all data at once instead of calculating each embedding during request time. The design choice removes the most resource-intensive task from every user's query because the internship dataset stays the same throughout server operation while the system uses permanent precomputed embeddings for server duration. FAISS performs complete dataset retrieval with exact results because it conducts inner product searches on L2-normalized vectors which yield similarity scores that only range from zero to one. The system selects exact search methods over approximate ones because the 595 record dataset size permits complete processing to finish within microseconds on typical CPU systems which makes approximation unnecessary while preserving complete retrieval accuracy. The system achieves its complete recommendation process from user skills entry to final result delivery in less than two seconds. The API provides five final recommendations through a structured JSON response which includes internship title, company name, domain, location, stipend, match score percentage, and direct apply link for each result that the frontend displays as interactive recommendation cards.

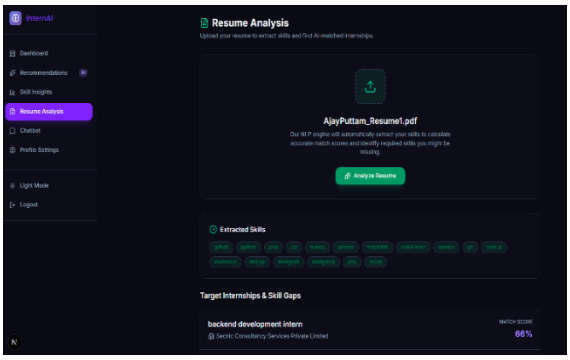

Fig. 4. Automatic technical skill extraction from an uploaded PDF resume using NLP-based keyword matching.

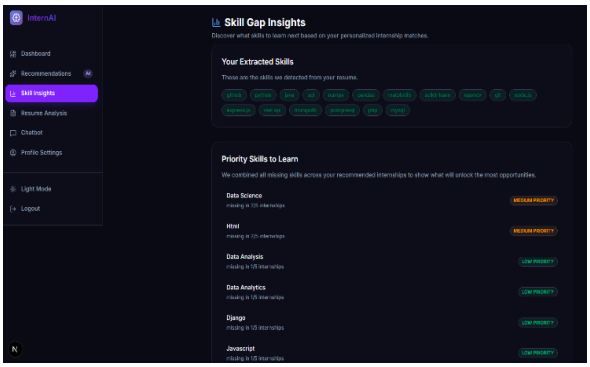

Fig. 5. AI-generated skill gap analysis and personalized learning roadmap produced for a recommended internship role.

VI. CONCLUSION

The article introduced an AI internship recommendation system which helps students eligible for the PM Internship Scheme to find their ideal internship positions without needing to conduct manual searches or use keyword filters. The system came into existence because students from rural and semi-urban areas who had the most restricted access to digital resources needed to solve the problem of their knowledge gaps with available educational resources. Students from smaller towns face challenges when they need to find jobs because they lack access to campus placement cells and structured career counselling systems which urban students use for their job search needs. The system design process used different access to career guidance, which people had, to create design choices that included a mobile-friendly interface and AI-generated learning roadmaps that required users to have no prior knowledge of career planning.

The approach used Sentence Transformer embeddings together with FAISS vector search to create a system which matched internship requirements with student profiles through semantic matching instead of matching exact words. The semantic approach enables the system to identify skill-based relationships which students possess because they have PyTorch proficiency, which allows them to apply for deep learning positions that require TensorFlow expertise. The four-component weighted ranking approach enabled recommendations to evaluate domain preference and region and experience level and skill alignment, which resulted in outcomes that matched the student's needs through both semantic and practical relevance.

The technique of soft domain diversity improved recommendation outcomes for various search queries because it prevented recommendations from being limited to one domain during open-ended search activities. The system provided essential support to students who had skills in multiple fields because it allowed them to receive five distinct recommendations instead of three which would have limited their ability to use the system effectively. The system uses a cumulative penalty that applies to repeated domains to force students to explore all relevant opportunities which improves their chances of discovering a perfect match.

The system maintained its performance level through all testing which included different query types and filter combinations and all domain sizes The edge case test with unrelated abilities correctly generated a no results response, indicating that the system manages both high-confidence and low-confidence cases appropriately. The mobile development question produced the strongest outcomes because it reached a final score of 0.559. The two-stage domain filter provided essential assistance for smaller domains which included data science and blockchain because strict matching alone proved insufficient.

The recommended system demonstrates superior performance compared to traditional internship search methods which depend on keyword matching while providing students with an effective yet simple solution for their PM Internship Scheme navigation needs. Research efforts will focus on expanding the dataset through additional domain integrations and developing direct connections to the official PM Internship portal which will enable real-time listing updates and users will use their feedback to enhance ranking performance over time. The proposed system creates a solid base which organizations can use to develop advanced career guidance systems that will benefit students throughout India.

A. Limitations

The evaluation results from assessment indicate that the system has different strengths which need to be acknowledged through transparent identification of its existing limitations. The recommendation quality depends on two factors which include the total number of internship listings in the 595-listing dataset and their age because the dataset only contains information from Internshala and Open Internships which was collected at one particular time. The dataset recommends static options because it does not update with new role additions or old role deletions which results in some recommendations showing roles that no longer accept applications. The domain expansion through two stages did not yield better recommendations for domains which have fewer than ten listings in the dataset that include blockchain, cybersecurity and game development. The reason for this is that the expansion process uses only a few related listings which do not exactly match the user's search intent.

B. Future Scope

The upcoming researchwork contains several important extensions which need to be studied. The permanent static dataset of 595 listings should be substituted by an active live data stream which connects to both the official PM Internship Scheme portal and the Internshala API to deliver current job recommendations. The system can create a user feedback loop which allows students to evaluate recommendation quality after which their feedback will be used to adjust ranking weights through human feedback-based reinforcement learning. The system can replace its dictionary-based skill extractor with a trained spaCy Named Entity Recognition model which uses technical resumes to identify both technical skills and soft skills along with emerging technologies. The implementation of a multilingual conversational interface with Groq and Llama 3 will enhance system accessibility for Indian students who do not speak English, which supports the inclusive objectives of the PM Internship Scheme.

REFERENCES

[1] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in *Proc. EMNLP-IJCNLP*, Hong Kong, 2019, pp. 3982–3992.

[2] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proc. NAACL-HLT*, Minneapolis, MN, USA, 2019, pp. 4171–4186.

[3] J. Johnson, M. Douze, and H. Jégou, "Billion-Scale Similarity Search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2021. [Online]. Available: https://github.com/facebookresearch/faiss

[4] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," in *Advances in Neural Information Processing Systems*, vol. 26, 2013, pp. 3111–3119.

[5] A. Dubey et al., "The Llama 3 Herd of Models," *arXiv preprint arXiv:2407.21783*, 2024. [Online]. Available: https://arxiv.org/abs/2407.21783

[6] Groq Inc., "Groq API Documentation," 2024. [Online]. Available: https://console.groq.com/docs . Accessed: Mar. 2026

[7] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.

[8] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W. Y. Ma, "Collaborative Knowledge Base Embedding for Recommender Systems," in *Proc. ACM SIGKDD*, San Francisco, CA, USA, 2016, pp. 353–362.

[9] P. Covington, J. Adams, and E. Sargin, "Deep Neural Networks for YouTube Recommendations," in *Proc. ACM RecSys*, Boston, MA, USA, 2016, pp. 191–198.

[10] C. Zhu, W. Zhu, F. Qing, and X. He, "Person-Job Fit: Adapting the Right Talent for the Right Job with Joint Representation Learning," in *Proc. KDD*, London, UK, 2018, pp. 2737–2745.

[11] L. Qin, Y. Chen, X. Zhu, and Y. Liu, "Enhanced Job Recommendation for Job Seekers," in *Proc. AAAI*, New Orleans, LA, USA, 2018, pp. 7814–7815.

[12] M. Jia, L. Xu, and F. Cao, "Skill Extraction from Job Postings Using Weak Supervision," in *Proc. ACL*, Dublin, Ireland, 2022, pp. 236–242.

[13] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*. Sebastopol, CA, USA: O'Reilly Media, 2009. [Online]. Available: https://www.nltk.org/book/

[14] S. Ramírez, "FastAPI: Modern, Fast Web Framework for Building APIs with Python," 2022. [Online]. Available: https://fastapi.tiangolo.com

[15] Vercel Inc., "Next.js Documentation - The React Framework for the Web," 2023. [Online]. Available: https://nextjs.org/docs