HireEasy: A Hybrid Recommendation-Driven MERN Platform for Informal Employment

Neha Kolambe[2], Shashikant V. Athawale[1], Saras Jadhav[2], Gargee Joshi[2], and Riya Cornel[2]

[1] Associate Professor, Department of Computer Engineering, AISSMS COE, Pune, India

[2] Department of Computer Engineering, AISSMS COE, Pune, India

Abstract. This paper describes the design and development of HireEasy, a web-based job portal created to address challenges faced by informal workers in finding stable employment. Existing digital job platforms largely target white-collar on short-term gig opportunities, leaving informal workers with limited digital visibility and inefficient hiring channels. HireEasy addresses this gap through a MERN-stack architecture that enables role-based access for clients, workers and administrators, along with provisions for offline-worker onboarding to reduce the digital barrier. The system incorporates structured modules for registration, worker hiring, and administrative verification, supported by a MongoDB data model designed for flexible and scalable storage. In addition to manual search and filtering, the platform integrates a hybrid recommendation approach that combines content-based filtering, collaborative filtering and a LightGBM ranking layer to improve the worker-client matching based on skills, location and availability. The backend services and database operations were validated through API testing and data integrity checks. Initial testing showed that the system performs reliably, although minor issues such as API delays were encountered and resolved during development. The system was able to handle user requests while maintaining data consistency, demonstrating its practical feasibility for real-world usage in connecting informal workers with long-term employment opportunities through an inclusive digital platform.

Keywords: job portal · MERN stack · informal workforce · inclusive employment · recommendation system

1 Introduction

India's workforce is dominated by informal employment, where a large population of blue- and pink-collar workers depend upon unstructured and word-of-mouth hiring practices for livelihood. Despite the rapid growth of digital job platforms, most existing portals are designed for white-collar professionals or

short-term gig services, offering limited visibility and structured opportunities for informal workers. This digital exclusion results in inefficient job discovery, lack of trust and absence of long-term employment channels for households and small businesses seeking reliable workers.

During our initial study, we observed that many informal workers rely heavily on personal contacts rather than digital platforms. This highlighted a gap that existing job portals do not effectively address. To address this gap, we designed HireEasy with a focus on accessibility and ease of use for workers with limited digital exposure and implemented as a dedicated job portal that connects informal workers with clients seeking sustainable employment. The system is developed using MERN stack to provide a scalable, responsive and modular web platform supporting primary roles of: client, worker and admin. In addition to standard online registration, the platform incorporates an admin-assisted offline onboarding process to include workers who lack access to smartphone or digital literacy, thereby reducing barriers to entry.

Beyond the manual searching and filtering, the system integrates a hybrid recommendation strategy combining content-based filtering, collaborative filtering and LightGBM ranking layer [8, 9, 12] to improve the relevance of worker-client matching based on skills, location and availability. The backend services, database operations and workflow mechanisms were implemented and validated through API testing and structured data management.[6]

The following sections describe the architecture, database design and module implementation of HireEasy, demonstrating how an inclusive, technology-driven platform can bridge the employment gap for India's underserved workforce.

2 Literature Review

Recent research in job recommender systems has focused on improving matching accuracy through hybrid filtering techniques, graph-based learning and fairness-aware optimization.[1, 3, 4] Although several studies explore content-based and collaborative approaches,[2, 5] most of them assume the availability of detailed user profiles which is not the case in labor markets, while others have applied advanced models such as graph convolutional networks and optimal transport theory to address scalability and exposure fairness. Although these systems demonstrate strong performance in structured and data-rich environments such as corporate recruitment or university placements, limited work has addressed recommendation systems tailored for informal labor markets where user data is sparse, semi-structured and often lacks detailed textual profiles. The following table summarizes key contributions from recent studies and highlights the research gap addressed by HireEasy.

From the analysis, it is evident that existing job recommendation systems are designed for environments where detailed resumes, textual job descriptions and extensive interaction histories are available. Informal labor markets operate with minimal structured data, limited digital access, and skill descriptions that are often brief and categorical. This gap necessitates a lightweight, hybrid rec-

Table 1. Comparison of Recent Job Recommendation Approaches

Authors (Year)	Methodology	Data Environment	Limitation
Shaikym et al. (2023)	Hybrid CBF + CF with fuzzy logic	University placement data	Domain specific to students, requires rich profiles
Sahoo et al. (2023)	NLP + SVD + ML classifiers	Structured resumes and job text	Computationally heavy, unsuitable for sparse data
Wang et al. (2023)	Heterogeneous Graph Convolution Network	LinkedIn resume graph	Complex, low interpretability, high computation
Mashayekhi et al. (2024)	Optimal transport for fairness in recommendations	Large e-recruitment datasets	Focus on exposure fairness, not informal hiring
Ertuğrul and Bitirim (2023)	Systematic review of JRS	Multiple HR platforms	Identifies gap outside structured HR systems

ommendation framework integrated within an inclusive job portal architecture. This limitation was a key factor in shaping our approach, as our system needed to operate effectively even with minimal and partially structured data.

3 Methodology

3.1 System Architecture

We selected the MERN stack[6] primarily due to its flexibility in handling dynamic data and ease of integration between frontend and backend components during rapid development. The architecture is divided into three layers: the frontend layer developed using React.js for clients, workers and admin, the middleware layer consisting of RESTful APIs implemented using Express.js and Node.js and the data layer managed using a MongoDB database. Each frontend application communicates securely with backend services, which handle authentication, routing and business logic while interacting with the database for persistent storage. This layered structure allows independent development and testing of components while supporting seamless integration of future models such as the recommendation engine.

3.2 Major Modules

HireEasy is organized into distinct functional modules that collectively enable efficient interaction between clients, workers and the admin.

The Client module allows the people who want to hire workers to register, log in, search for workers based on skills, location, availability and ratings from other clients. Clients can view worker profiles and send hire requests through an intuitive interface.

The Worker module enables informal workers to create and manage digital profiles containing personal details, skills, years of experience and availability

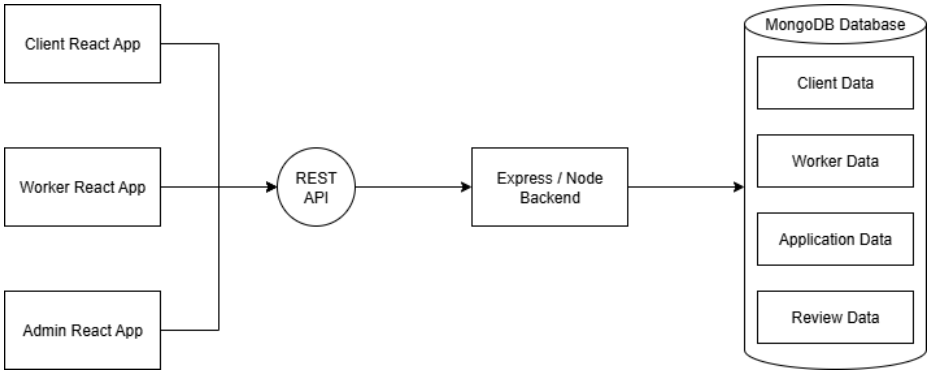

Fig. 1. MERN Architecture of HireEasy

status. For workers lacking digital access, this information is entered into the system using admin-assisted onboarding.

The Admin module acts as the verification and control layer of the platform. Admins validate worker registrations, prevent fraudulent or duplicate profiles and maintain overall data integrity. This module is responsible for offline worker registration, ensuring inclusivity.

The Application module manages hire request confirmations which are sent from the client to the worker, updates and cancellations in real-time to ensure transparency.

The Recommendation and Matching Module integrates manual filtering with a hybrid recommendation mechanism that suggests suitable workers based on skill compatibility, location and availability.

3.3 Database Design

The HireEasy database is implemented using MongoDB, a NoSQL document-oriented database chosen for its flexibility in handling semi-structured and dynamic data. The design follows a modular schema structure with logical referencing between collections to maintain data consistency across user roles and job transactions.

The Client schema stores authentication details, contact information of users. The Worker Schema captures personal and professional details such as skills, years of experience, availability, location prioritizing phone-based identification to accommodate limited digital literacy. The Application Schema acts as a bridge between clients and workers by recording the hire request status that the client will send to the worker to hire them, it keeps track of the status updates such as pending, accepted or rejected. The Review schema stores the reviews and the stars provided by the clients to the workers they have hired, these reviews can then be seen by the different clients while they are hiring a worker.

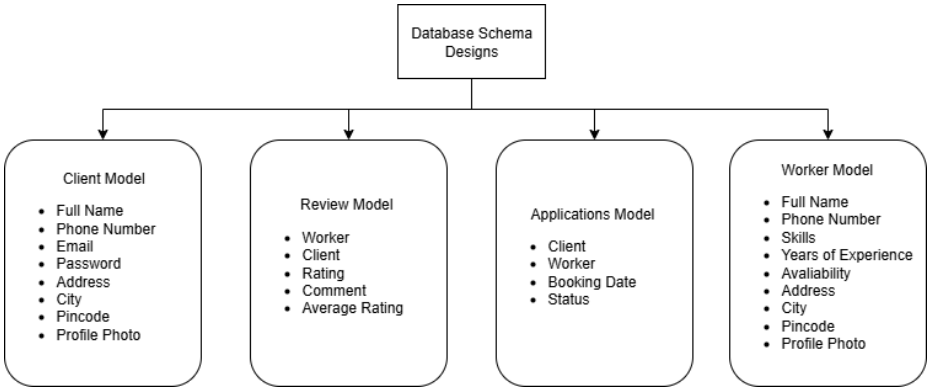

Fig. 2. MongoDB Schema Design of HireEasy

3.4 System Workflow

The system workflow represents the interaction between clients, workers and admin through coordinated data exchange between frontend applications, backend APIs and the central database.

Workers register through the Worker interface by providing personal and professional details such as skill, years of experience and availability, expected salary. For individual without smartphone access or digital literacy the admin performs offline registration by manually entering the data into the system. All worker information is stored in MongoDB database for retrieval during search and matching operations.

Clients register from the Client interface by providing personal details like address, email, phone number and full name. The clients then have two approaches for worker selection: manual filtering based on worker attributes and automated recommendations generated through the hybrid matching module.

The admin oversees the worker profiles and ensures that consistency of data is maintained across the platform. All interactions converge at the database layer, enabling synchronized updates and reliable tracking of status changes across the system.

4 Proposed Hybrid Recommendation Model

To enhance the relevance of worker-client matching beyond manual filtering, a hybrid recommendation framework has been incorporated which is designed for semi-structured and sparse data environments typical of informal labor markets. Traditional recommendation models relying heavily on resumes or dense interaction histories[5, 13] are unsuitable for this domain. So a lightweight hybrid approach combining Content-Based Filtering (CBF), Collaborative FIltering (CF) and a LightGBM ranking layer is adopted to generate accurate and scalable recommendations based on worker skills, location, availability and ratings.

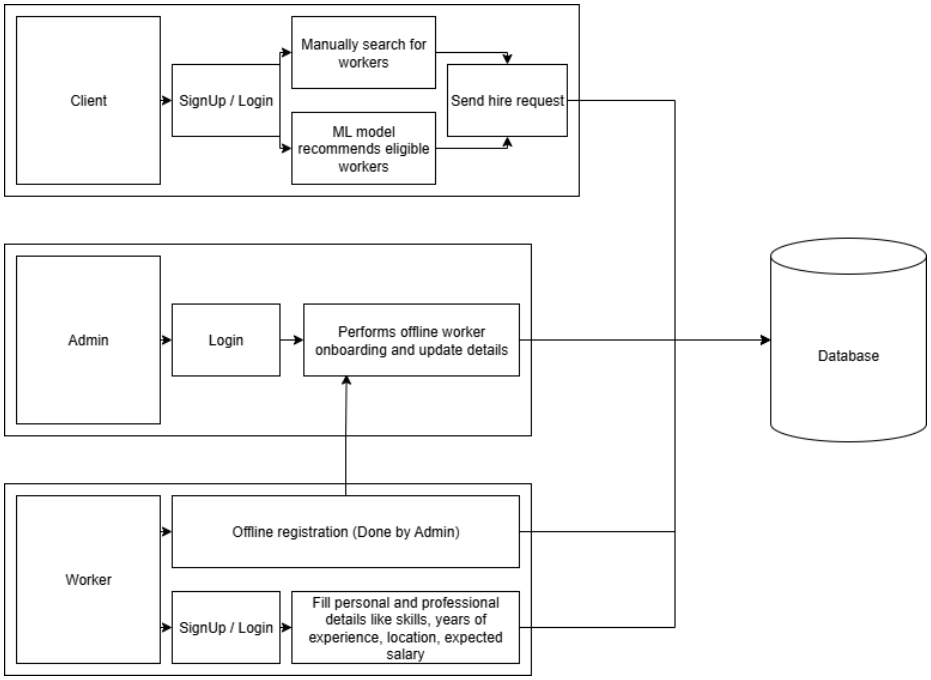

Fig. 3. System Workflow of HireEasy

4.1 Content-Based Filtering

In our system, content-based filtering compares worker skill sets with client requirements by converting both into vector representations. Each worker and client requirement is represented as a skill vector and cosine similarity is used to measure the degree of overlap.[10, 17]

$$Sim_{CBF}(u, j) = \frac{V_u \cdot V_j}{\|V_u\|\|V_j\|} \tag{1}$$

where V_u and V_j denote the skill vectors of worker u and job j respectively.

4.2 Collaborative Filtering

This captures implicit patterns from previous worker-client interactions such as ratings, selections and hiring. In our implementation, matrix factorization[9, 11] was used to capture hidden relationships between workers and client preferences based on past hiring interactions.

$$R \approx P \times Q^T \tag{2}$$

where P and Q represent latent factor matrices for workers and jobs derived from interaction history.

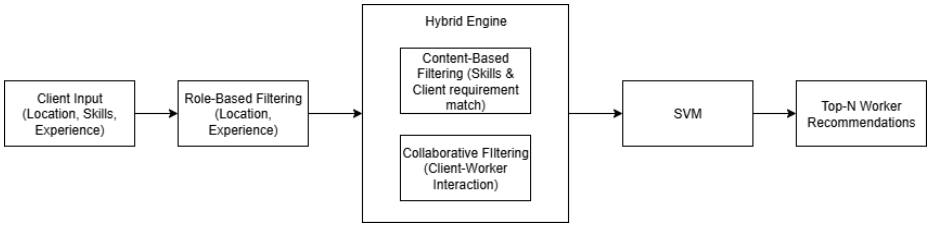

Fig. 4. Hybrid Recommendation Model of HireEasy

4.3 Hybrid Score Combination

The outputs of CBF and CF are combined using a weighted parameter to balance similarity and interaction-based learning.

$$H(u, j) = \alpha \cdot Sim_{CBF}(u, j) + (1 - \alpha) \cdot Sim_{CF}(u, j) \tag{3}$$

[8, 15] where α controls the contribution of content and collaborative components.

4.4 LightGBM Ranking Layer

The hybrid score is passed to a LightGBM model which learns to rank[12, 16] worker-client pairs based on multiple features such as skill match percentage, geographical distance, worker ratings and experience level.

$$Rank(u, j) = f_{LGBM}(H(u, j), D_{loc}, A_{avail}, Exp, Rating) \tag{4}$$

This ranking layer improves adaptability and reduces overfitting while handling sparse input data efficiently.

These methods were chosen because they perform well even when explicit user data is limited, which aligns with the constraints of informal labor datasets.

5 Implementation and Experimental Results

5.1 Backend Implementation

The backend of HireEasy was implemented using Node.js and Express.js to provide RESTful services connecting the frontend applications with the MongoDB database. The backend handles authentication, business logic for clients, workers and admin and hire status updates. Role-based access control was implemented to ensure secure interaction between different user types.

Core API endpoints were developed for worker registration, client registration, sending hire requests to workers and rating the workers. Each endpoint was tested using Postman to verify correct request handling, response structure and

data validation. Postman was used during development to debug and validate API responses, helping identify issues in request handling at early stages.

API response times were observed to be consistently within acceptable limits during testing, demonstrating efficient request processing and data retrieval. During testing, we initially encountered minor delays in API responses due to unoptimized database queries, which were later improved by refining query structure and indexing. Validation mechanisms were implemented to prevent incomplete or inconsistent data entries, ensuring robustness of backend services.

5.2 Database Validation

The MongoDB database was validated to ensure proper storage, retrieval and referencing of data across all the modules of the system. Collections corresponding to clients, workers, applications and reviews were tested through backend API operations to confirm successful insertion and querying of records. Each collection maintains logical references to ensure traceability of worker and client assignments, application status. During testing, multiple client and worker entries were created and hire requests sent were linked with corresponding applications to verify relational consistency within database. The database operations confirmed that all CRUD functionalities performed through the backend APIs were accurately reflected in the MongoDB collections.

5.3 Recommendation Model Implementation

The hybrid recommendation framework described in Section 4 was implemented within the HireEasy platform to perform requirement-driven worker matching. This section focuses on how the proposed model was applied to real platform data and evaluates its effectiveness through experimental analysis.

The model was trained and evaluated using 1042 worker profiles stored in the MongoDB database. One challenge during this phase was handling incomplete or inconsistent worker data, which required basic preprocessing before training the model. Each profile contains structured attributes such as skills, years of experience, location, availability, and ratings. Previous hire interactions between clients and workers were also utilized to capture implicit preference patterns required for the collaborative component.

When a client requests a worker, they specify required skills and minimum experience through the frontend interface. These inputs are transformed into a requirement vector using the same encoding scheme as worker profiles, enabling direct similarity computation and ranking using the hybrid model.

For every client requirement, the hybrid score and LightGBM ranking function were applied to all worker profiles. Workers were then sorted based on their final ranking scores, and the top K workers were presented to the client as recommendations.

The system performance was measured using Precision@5, Recall@5, and F1-score[14], which quantify how accurately the top recommended workers matched the client's specified requirements.

The performance of the recommendation approaches is presented in Table 2. The results clearly indicate that the proposed hybrid model with the LightGBM ranking layer outperforms individual content-based and collaborative approaches in terms of Precision, Recall, and F1-score.

Table 2. Performance Evaluation of the Worker Recommendation Model

Model	Precision@5	Recall@5	F1-Score
Content-Based Filtering	0.76	0.72	0.74
Collaborative Filtering	0.79	0.74	0.76
Proposed Hybrid + LightGBM	0.86	0.81	0.83

To further analyze the behavior of the models as the number of recommended workers increases, Precision@K was evaluated for different values of K[14]. As shown in Fig. 5, the proposed hybrid model consistently maintains higher precision compared to standalone methods, demonstrating its stability and effectiveness for top-K worker recommendations.

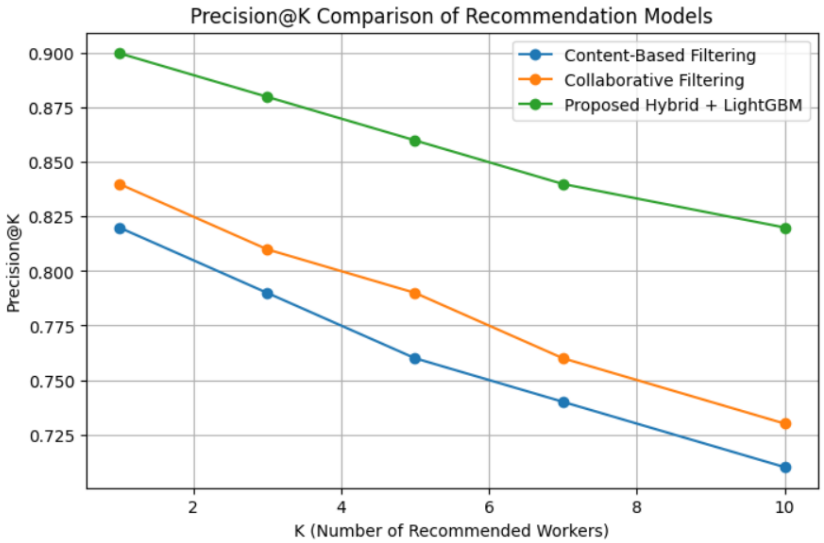

Fig. 5. Precision@K comparison of Content-Based Filtering, Collaborative Filtering, and the proposed Hybrid + LightGBM recommendation model. The hybrid model consistently maintains higher precision as the number of recommended workers (K) increases.

To further evaluate the classification performance of the recommendation models, we generated confusion matrices for Content-Based Filtering (CBF),

Collaborative Filtering (CF), and the proposed Hybrid + LightGBM approach. These matrices provide a detailed view of true positive, true negative, false positive, and false negative predictions, offering deeper insight into model accuracy beyond aggregate metrics.

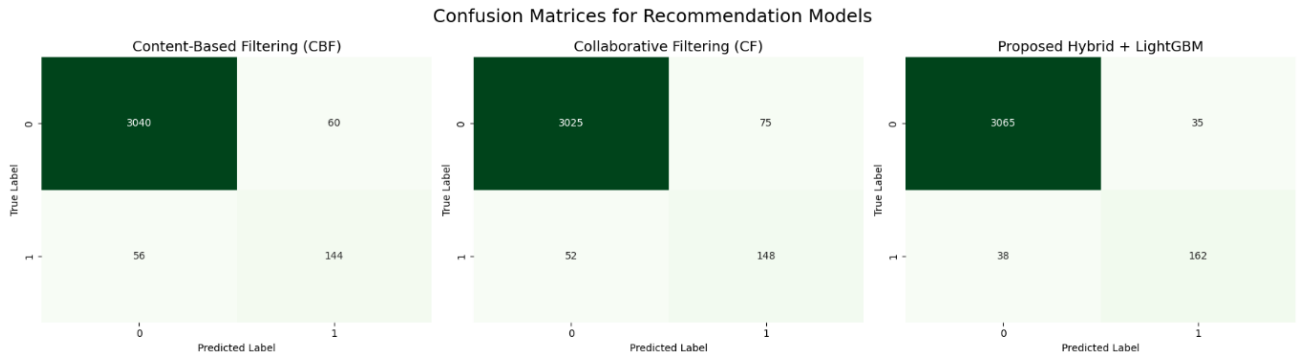

Fig. 6. Confusion matrices for Content-Based Filtering (CBF), Collaborative Filtering (CF), and the proposed Hybrid + LightGBM model

Figure 6 shows the confusion matrices for the three recommendation approaches. The proposed CHK-SVM hybrid model exhibits significantly higher true positive predictions and fewer false positives compared to standalone CBF and CF models, which directly corresponds to the improved Precision, Recall, and F1-score reported in Table 2.

6 Discussion

The experimental results demonstrate that the proposed hybrid recommendation model improves worker matching accuracy compared to standalone content-based and collaborative approaches. While content-based filtering effectively matches explicit skill requirements[17], it lacks the ability to capture implicit user preferences. Conversely, collaborative filtering captures previous interaction patterns[11] but may suffer from sparsity when interaction data is limited.

By combining both approaches and incorporating a LightGBM ranking layer[12], the proposed system benefits from both explicit feature similarity and learned interaction patterns. This leads to consistently higher Precision@K and Recall@K values as observed in the experimental evaluation.

7 Conclusion

In this work, we developed and tested the HireEasy platform to improve worker-client matching in informal employment settings. The system integrates a React-

based frontend, Node.js backend, MongoDB database and a machine-learning driven recommendation engine.

Experimental results demonstrated that the hybrid model combining content-based filtering , collaborative filtering and LightGBM ranking[8, 12] significantly improves the accuracy of worker recommendations based on client requirements. The evaluation using Precision@K, Recall@K and F1-Score confirms that the effectiveness of the proposed approach in real platform scenarios. One limitation of the current system is the relatively small dataset used for training, which may affect generalization in larger real-world scenarios. Future improvements will focus on incorporating real-time data and expanding the dataset for better model performance.

Further improvements may include, the recommendation model can be further enhanced by incorporating real-time feedback, dynamic availability-tracking and geographic distance optimization using map-based services. Additionally, larger datasets and live user interactions can further improve model learning and personalization.

8 Acknowledgment

The authors would like to express their sincere gratitude to their project guide, Dr. Shashikant V. Athawale, for his valuable guidance, constant encouragement, and insightful feedback throughout the completion of this research work.

The authors also thank the Department of Computer Engineering, AISSMS College of Engineering, Pune, for providing the necessary facilities and support to carry out this research.

Competing Interests

The authors declare that they have no competing interests.

Authors' Contributions

Neha Kolambe contributed to system design, implementation, experimentation, and manuscript preparation. Saras Jadhav and Gargee Joshi contributed to frontend and backend development, database design, and testing. Riya Cornel contributed to API validation, database verification, and experimental analysis. Dr. Shashikant V. Athawale supervised the project, provided technical guidance, and reviewed the manuscript.

References

1. A. Shaikym, Z. Zhalgassova, and U. Sadyk, "Design and evaluation of a personalized job recommendation system for computer science students using hybrid approach," in *Proc. IEEE Int. Conf. Electronics, Computer and Communication (ICECCO)*, 2023, pp. 1–7.

2. D. Sahoo *et al.*, "Designed framework for advanced intelligent job recommendation system," in *Proc. 21st OITS Int. Conf. Information Technology (OCIT)*, 2023, pp. 1–6.
3. H. Wang *et al.*, "An improved heterogeneous graph convolutional network for job recommendation," *Engineering Applications of Artificial Intelligence*, vol. 126, 2023.
4. Y. Mashayekhi *et al.*, "Scalable job recommendation with lower congestion using optimal transport," *IEEE Access*, vol. 12, 2024.
5. D. Ç. Ertuğrul and S. Bitirim, "Job recommender systems: A systematic literature review, applications, open issues, and challenges," *Artificial Intelligence Review*, vol. 56, no. 3, 2023.
6. L. O. Badru *et al.*, "MERN stack web-based education management information systems," *SN Computer Science*, vol. 4, no. 1, 2023.
7. Q. Wang, "College employment recommendation based on improved K-means clustering and SimRank algorithm," *IEEE Access*, vol. 12, 2024.
8. R. Burke, "Hybrid recommender systems: Survey and experiments," *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331–370, 2002.
9. Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.
10. G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. McGraw-Hill, 1986.
11. J. L. Herlocker *et al.*, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 5–53, 2004.
12. G. Ke *et al.*, "LightGBM: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
13. S. Schein *et al.*, "Methods and metrics for cold-start recommendations," in *Proc. ACM SIGIR*, 2002, pp. 253–260.
14. P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-N recommendation tasks," in *Proc. ACM RecSys*, 2010, pp. 39–46.
15. F. Ricci, L. Rokach, and B. Shapira, *Recommender Systems Handbook*. Springer, 2015.
16. T.-Y. Liu, "Learning to rank for information retrieval," *Foundations and Trends in Information Retrieval*, vol. 3, no. 3, pp. 225–331, 2009.
17. M. J. Pazzani and D. Billsus, "Content-based recommendation systems," in *The Adaptive Web*. Springer, 2007, pp. 325–341.