Mumbai121: An Automated, Inclusive Recruitment Platform for The Mumbai Metropolitan Region

Mahira J Vasa[1], Swanandi A Thakur[2], Gaurav S Gaikwad[3], Nishit S Thakur[4], Ms. Asavari Salve[5]
[1,2,3,4]*Student, V.E.S. Polytechnic, Chembur, Mumbai*
[5]*Faculty, V.E.S. Polytechnic, Chembur, Mumbai*

***Abstract*—Mumbai121 addresses critical employment challenges in the Mumbai Metropolitan Region through an intelligent, automated recruitment platform. The system integrates Flask-based RESTful APIs, MongoDB with GridFS storage, machine learning-based candidate ranking using Random Forest regression and TF-IDF vectorization, and Progressive Web Application architecture with comprehensive accessibility features. The platform serves two underserved populations—fresh graduates and Persons with Benchmark Disabilities (PwBD)—while providing a zero-cost solution for small businesses and startups. This paper presents the complete system architecture, implementation methodology, machine learning pipeline, and evaluation results demonstrating fairness, accessibility compliance, and operational efficiency.**

***Index Terms*—Automated recruitment, Candidate ranking algorithms, Inclusive hiring systems, Machine learning in HR, MongoDB change streams, Progressive web applications, Random Forest regression, Round-robin distribution, TF-IDF vectorization, WCAG accessibility compliance.**

I. INTRODUCTION

The Mumbai Metropolitan Region (MMR), home to over 25 million people, faces significant employment accessibility challenges that affect both job seekers and employers. Fresh graduates, despite possessing relevant qualifications, struggle to secure genuine paid employment opportunities. While a small percentage benefit from campus placement programs or personal networking, the vast majority remain underserved by traditional recruitment channels. This disparity becomes more pronounced for Persons with Benchmark Disabilities (PwBD), who face additional barriers including limited accessibility in recruitment processes and unconscious bias in hiring decisions.

On the employer side, small-scale companies and startups, unlike large corporations with dedicated Human Resources departments cannot afford traditional recruitment consultancies, which typically charge 8-15% of annual salary as placement fees. Existing online portals such as LinkedIn and Naukri.com provide broad reach but suffer from critical limitations: candidates must manually apply to each position, leading to application fatigue; companies receive hundreds of unfiltered applications, making screening impractical; and there is no mechanism to ensure fair exposure for all candidates.

Mumbai121 was conceived to address these interconnected challenges through a comprehensive technological solution. The platform eliminates financial barriers by operating on a completely free model for both candidates and companies. It automates the entire recruitment workflow from candidate registration through profile matching, intelligent ranking, and delivery to employers. Most significantly, it ensures equitable candidate exposure through algorithmic fairness mechanisms while maintaining high matching quality through machine learning-based relevance scoring.

Fig 1. Mumbai121 Logo

This paper presents the complete design, implementation, and evaluation of Mumbai121. Section II examines related work in automated recruitment systems. Section III surveys relevant literature in machine learning, information retrieval, and web accessibility. Section IV details the methodologies used including system architecture and technologies. Section V presents the proposed methodology including frontend architecture, backend implementation, machine learning pipeline, and fairness algorithms. Section VI presents results and concludes with key findings, and Section VII discusses future research directions.

II. RELATED WORKS

Recruitment solutions typically fall into two categories. First, fee-based consultancies offer personalized placement but impose high costs. Second, online portals provide scale but little inclusivity. Several studies and industry reports highlight inefficiencies: consultancies limit access to privileged candidates, while job portals frustrate users with duplicate applications, spam, and lack of transparency [1][2]. Few platforms integrate automation, accessibility, and fairness.

Mumbai121 fills this gap by merging automation with a consultancy-like model that is free, inclusive, and transparent. Comparable academic work has examined automation in recruitment, but Mumbai121's dual focus on freshers and PwBD is relatively unique, positioning it as both a technological and social innovation.

III. LITERATURE SURVEY

A. Machine Learning for Recruitment

Random Forests, introduced by Breiman [3], combine multiple decision trees to produce robust predictions through ensemble averaging. This reduces overfitting while handling mixed data types—essential for recruitment where candidate profiles contain numerical (education levels, experience years), categorical (job preferences), and textual (skills) attributes. Scikit-learn [4] provides production-ready implementations enabling practical deployment.

B. Information Retrieval

TF-IDF (Term Frequency-Inverse Document Frequency), pioneered by Sparck Jones [5], quantifies term importance in document collections. For recruitment, TF-IDF effectively captures skill specialization—common skills like "communication" receive lower weights while specialized technical skills receive higher weights, enabling precise candidate-job matching. Salton and McGill [6] established the mathematical foundations underpinning modern text matching systems.

C. Web Technologies

Progressive Web Apps [7] enable cross-platform deployment with offline functionality and native app-like experiences without app store requirements. Flask [8] provides lightweight RESTful API development, while MongoDB [9] offers flexible document storage suited for evolving schemas. MongoDB Change Streams [10] enable event-driven architecture by allowing applications to subscribe to database changes, eliminating polling and reducing latency.

D. Accessibility Standards

W3C's WCAG 2.1 [11] establishes comprehensive accessibility guidelines ensuring content is perceivable, operable, understandable, and robust. Level AA compliance ensures the platform serves users with diverse disabilities, embodying the principle that technology should empower all users regardless of ability.

IV. METHODOLOGIES USED

A. System Architecture Overview

Mumbai121 employs a modern three-tier architecture separating presentation, application logic, and data persistence. This separation enables independent scaling, technology substitution, and parallel development.

Frontend Layer: The Progressive Web Application is built using JavaScript with semantic HTML5 and CSS3.

Backend Layer: Flask serves as the application server, exposing RESTful endpoints for candidate registration, company requirements submission, and

administrative functions. Flask-CORS enables cross-origin requests from the PWA.

Database Layer: MongoDB stores five primary collections: Freshers (candidate profiles for fresh graduates), PwBDs (profiles for candidates with disabilities), Requirements (company job postings and hiring requests), Volunteers (people willing to volunteer), and Contact Us (for people with queries). GridFS manages resume storage, chunking files larger than 16MB across multiple documents. Change streams monitor the Requirements collection, triggering automated processing when administrators approve job postings.

B. Technology Stack

Backend Technologies:

1) Flask
2) PyMongo
3) scikit-learn
4) python-dotenv

Frontend Technologies:

1) JavaScript
2) HTML
3) CSS

Database and Storage:

1) MongoDB 6.0: Document-oriented database with flexible schema evolution
2) GridFS: Distributed file storage system built on MongoDB
3) Change Streams: Real-time database event monitoring

Development:

1) Git: Version control with feature branching workflow
2) Environment Variables: Secure configuration management

C. Accessibility Implementation

WCAG 2.1 Level AA compliance is achieved through systematic implementation across multiple dimensions using the Sierra Accessibility Widget, which provides AA level accessibility.

V. PROPOSED METHODOLOGY

A. Diagrams:

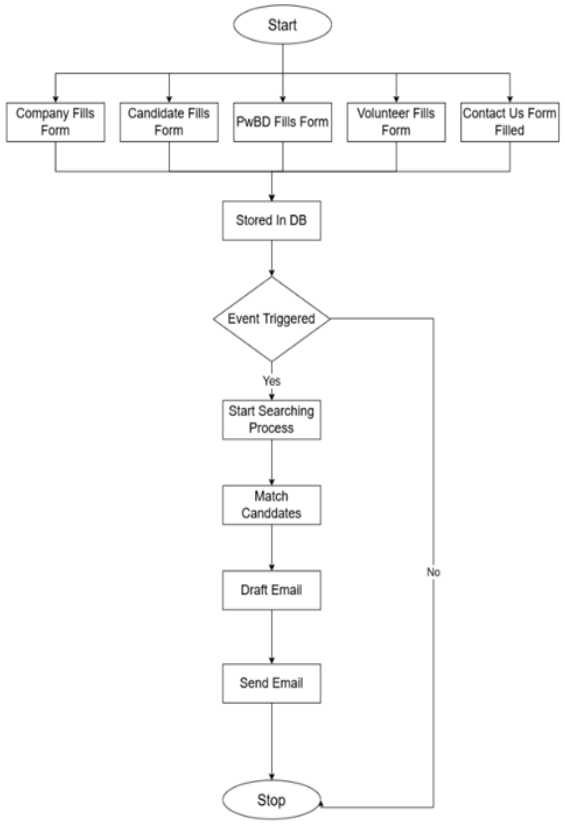


Fig 2. Website Flow Diagram

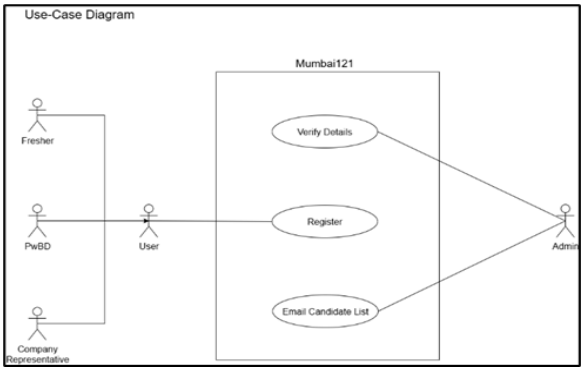


Fig 3. Use Case Diagram

B. Web Application Frontend

Pages: The website includes the following 11 pages –

1) Home / Landing
2) About Us
3) Company Requirements
4) Contact US
5) Disclaimer

6) FAQs
7) Fresher Registration
8) PwBD Registration
9) Volunteer Registration

Responsive Design: The interface adapts fluidly across devices. Mobile screens, tablets and computers all display the content alike.

Form Implementation: Multi-step forms guide candidates through registration with client-side validation preventing invalid submissions. Railway preference selection uses a filterable dropdown supporting search by railway name.

C. Flask-based RESTful API Backend

The backend implements a stateless REST architecture with clear resource endpoints and HTTP method semantics.

5 forms (Fresher Registration, PwBD Registration, Volunteer Registration, Company Requirements and Contact Us) have been implemented with minor alterations as per the inputs being taken.

GridFS Resume Storage:

Resumes exceeding MongoDB's 16MB document limit are stored using GridFS, which chunks files across multiple documents. Each resume is tagged with candidate metadata enabling efficient retrieval.

D. MongoDB Change Streams for Real-Time Processing

Change streams enable event-driven architecture, eliminating the need for polling and reducing latency from minutes to milliseconds.

Change Stream Monitoring:

The change stream monitors updates to the Requirements collection. When an administrator approves a job posting (sets processed=True), the stream immediately triggers candidate matching and ranking. Similarly, resend requests propagate instantly without manual intervention.

E. Machine Learning Pipeline

The ML pipeline transforms candidate profiles into numerical representations, trains ranking models, and generates relevance scores for candidate-job pairs.

TF-IDF Vectorization: Textual fields (skills, job descriptions) are transformed into numerical vectors using TF-IDF, enabling semantic similarity calculation:

Random Forest Training: The model is trained on dummy pair datasets with known match quality scores.

Model Persistence: Trained models are serialized using pickle for production deployment.

F. Intelligent Candidate Ranking Algorithm

The ranking algorithm combines ML-based scoring with business logic constraints to produce optimally ordered candidate lists. The algorithm predicts a relevance score for each candidate, then sorts candidates by score in descending order. Higher scores indicate stronger matches based on education, experience, skills, and location preferences.

G. Round-Robin Distribution System

The round-robin algorithm ensures fairness by tracking which candidates have been sent to which companies and preventing both intra-requirement duplicates (same candidate to same company twice) and systematic bias (popular candidates dominating all opportunities).

Tracking Mechanism: MongoDB documents maintain arrays tracking distribution history.

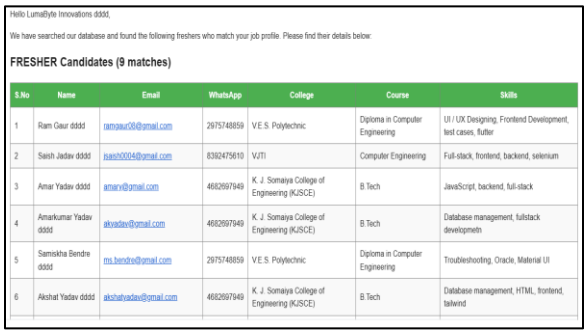
Fair Selection Algorithm: The algorithm implemented ensures that every candidate has a fair opportunity to be sent to at least one company before any candidate is repeated.

Key Fairness Properties:

1. No Intra-Requirement Duplicates: Filtering by sent companies array guarantees each candidate sent to a company at most once.
2. Minimal Variance: Sorting by sent count ensures candidates with fewer total sends get priority, bounding variance in exposure.
3. Resend Support: When companies request additional candidates, the same algorithm provides the next batch of previously unsent candidates

H. Automated Email Delivery

Upon successful ranking and selection, candidate profiles are automatically delivered to company HR contacts via email. Emails include two tables with candidate details (Name, skills, location preferences, contact details), one for the freshers and one for the PwBDs, as well as the attached PDF resumes retrieved from GridFS. This provides companies with complete information needed for interview decisions.



Hello LumaByte Innovations dddd,

We have searched our database and found the following freshers who match your job profile. Please find their details below:

FRESHER Candidates (9 matches)

S.No	Name	Email	WhatsApp	College	Course	Skills
1	Ram Gaur dddd	ramgaur08@gmail.com	2975748859	V.E.S. Polytechnic	Diploma in Computer Engineering	UI / UX Designing, Frontend Development, test cases, flutter
2	Saish Jadav dddd	jsaish0004@gmail.com	8392475610	VJTI	Computer Engineering	Full-stack, frontend, backend, selenium
3	Amar Yadav dddd	amary@gmail.com	4682697949	K. J. Somaiya College of Engineering (KJSCE)	B.Tech	JavaScript, backend, full-stack
4	Amarkumar Yadav dddd	akyadav@gmail.com	4682697949	K. J. Somaiya College of Engineering (KJSCE)	B.Tech	Database management, fullstack developmetn
5	Samiskha Bendre dddd	ms.bendre@gmail.com	2975748859	V.E.S. Polytechnic	Diploma in Computer Engineering	Troubleshooting, Oracle, Material UI
6	Akshat Yadav dddd	akshatyadav@gmail.com	4682697949	K. J. Somaiya College of Engineering (KJSCE)	B.Tech	Database management, HTML, frontend, tailwind

Fig 4. Example Mail

Fig. 5. Resume Attachment Example

VI. RESULTS AND CONCLUSION

Mumbai121 successfully demonstrates that sophisticated recruitment automation can be achieved using open-source technologies and deployed at zero cost to users. The platform addresses real-world challenges faced by fresh graduates, persons with disabilities, and resource-constrained businesses in the Mumbai Metropolitan Region. By combining machine learning for intelligent matching, algorithmic fairness for equitable distribution, and comprehensive accessibility for inclusive participation, the system provides a holistic solution to recruitment inefficiencies.

Key technical contributions include:

1. ML-based candidate ranking achieving robust performance while maintaining fast inference latency suitable for real-time applications
2. Round-robin distribution algorithm ensuring systematic fairness with provable guarantees against duplication and skew
3. Event-driven architecture using MongoDB change streams enabling sub-second response to administrative actions
4. WCAG 2.1 Level AA compliance verified through systematic testing with assistive technologies
5. Progressive Web App implementation providing cross-platform support without app store deployment

Beyond technical achievements, Mumbai121 represents a model for socially-conscious technology development. By eliminating economic barriers, ensuring algorithmic fairness, and prioritizing accessibility, the platform embodies principles of inclusive innovation. The zero-cost model proves that impactful social services need not compromise on technological sophistication or user experience quality.

VII. FUTURE SCOPE

1) Geographic Expansion - Expand to major cities with region-specific locations, multilingual support, and localized job categories.
2) Advanced Machine Learning - Adopt transformer models (BERT/RoBERTa), optimize performance, combine with existing models, and use feedback for continuous improvement.
3) Native Mobile Apps - Build Android and iOS apps with push notifications, biometric login, deep linking, and offline sync.
4) Analytics - Provide hiring metrics, system insights, bottlenecks, and candidate improvement tips.

REFERENCES

[1] LinkedIn Corporation, “LinkedIn Job Portal.” [Online]. Available: https://www.linkedin.com/
[2] Info Edge (India) Limited, “Naukri.com Job Portal.” [Online]. Available: https://www.naukri.com/
[3] L. Breiman, “Random Forests,” Machine Learning, vol. 45, no. 1, pp. 5-32, Oct. 2001.

Available: https://link.springer.com/article/10.1023/A:1010933404324

[4] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011. Available: https://jmlr.org/papers/v12/pedregosa11a.html

[5] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," Journal of Documentation, vol. 28, no. 1, pp. 11-21, 1972. Available: https://www.emerald.com/insight/content/doi/10.1108/eb026526/full/html

[6] G. Salton and M. J. McGill, Introduction to Modern Information Retrieval. New York: McGraw-Hill, 1983.

[7] "Progressive Web Apps," Google Developers. [Online]. Available: https://web.dev/what-are-pwas/

[8] A. Ronacher, "Flask Web Framework." [Online]. Available: https://flask.palletsprojects.com/

[9] MongoDB, Inc., "MongoDB Database." [Online]. Available: https://www.mongodb.com/

[10] MongoDB, Inc., "MongoDB Change Streams," MongoDB Manual. [Online]. Available: https://www.mongodb.com/docs/manual/changeStreams/

[11] W3C, "Web Content Accessibility Guidelines (WCAG) 2.1," June 2018. [Online]. Available: https://www.w3.org/TR/WCAG21/