A Web-Based Recruitment System Using Java Technologies

Nalawade Ishwari Atul[1], Awari Kanchan Santosh[2], Mane Shradha Sunil[3], Sawant Uddhav Chandu[4], V P Tonde[5]
[5]Assistant professor
[1, 2, 3, 4, 5] Sinhgad Institute of Technology Lonavala, India

Abstract- *This paper describes the design and development of a web-based Job Portal System aimed at making the recruitment process faster and less cumbersome. Built with Java, Servlets, JSP, and MySQL, the platform brings job seekers and employers together in one place. Candidates can register, build profiles, upload their resumes, and submit applications, while employers get tools to post openings and track incoming applicants. Rather than relying on AI-driven matching, the system uses skill-, experience-, and category-based database filtering to connect the right candidates with the right roles. Security, clean data flow, and ease of use were central priorities throughout development. Evaluation results confirm that the system meaningfully cuts down recruitment time, lowers manual workload, and makes the hiring process more accessible for everyone involved.*

Keywords: Job Portal System, Online Recruitment, Java, JSP, Servlets, MySQL, Web Application, Resume Management, Automated Hiring

I. INTRODUCTION

The way organizations find and hire talent has changed dramatically over the past two decades. Before the internet became widely accessible, recruitment largely depended on newspaper classifieds, walk-in drives, campus visits, and employment agencies. These approaches demanded heavy manual involvement — sorting paper resumes, making individual phone calls, and coordinating through letters — all of which made hiring slow, expensive, and geographically confined. A small business in one city had little realistic way to reach a qualified candidate in another.

Digital communication changed that equation entirely. Web-based job portals created a shared space where candidates can browse openings and submit applications from anywhere, and employers can manage the entire hiring pipeline without being physically present. Geographic barriers dropped, response times shortened, and the sheer volume of reachable applicants grew substantially.

The system described in this paper is built using Java Servlets and JavaServer Pages (JSP) on the front and middle tiers, with MySQL handling data storage at the back. Java was chosen deliberately — it is well-regarded for building secure, scalable, and platform-independent applications, qualities that matter in a recruitment context where data sensitivity and uptime are both important. The platform covers all the core functions a recruitment system needs: user registration and login, profile and resume management, job posting, application submission, and employer-side candidate tracking.

One deliberate design choice worth highlighting is the avoidance of machine learning or AI-based matching. Instead, the system uses SQL query filtering — candidates see jobs that match their listed skills, qualifications, and experience level based on structured database lookups. This keeps the system straightforward to maintain and audit, while still producing relevant results. Employers, in turn, can search the candidate pool using criteria specific to their open roles.

Security receives dedicated attention throughout. Session management, input validation, and authentication controls are built in to guard against unauthorized access and keep data clean. The modular architecture also leaves the door open for future upgrades — email notifications, a recommendation engine, mobile access, or analytics dashboards could all be layered in without overhauling the core system.

The broader goal is a platform that genuinely simplifies hiring for both sides of the table, proving that well-applied standard web technologies can solve real recruitment problems without unnecessary complexity.

II. MAIN OBJECTIVE

The core aim of this project is to build a web-based recruitment platform that makes the hiring process smoother and more direct for both candidates and employers. For job seekers, the system offers a space to register, build out a profile, upload a resume, and apply for roles that fit their back-ground. For employers, it provides tools to publish vacancies, review incoming applications, and identify strong candidates without wading through unnecessary steps.

Beyond convenience, the project also targets the ineffi-ciencies baked into traditional hiring — manual data entry, paper-based tracking, and time-consuming application sorting. Automating these tasks frees up time on both sides. Security is another priority; the system includes authentication and input validation so that user data is handled responsibly.

Together, these elements aim to deliver a recruitment tool that is efficient, trustworthy, and built to grow.

III. RELATED WORKS

Interest in digital recruitment platforms has grown steadily as organizations look for faster, more scalable alternatives to traditional hiring. Early systems were fairly basic — job listings posted online with email-based application submis-sion — but they demonstrated a clear appetite for moving recruitment off paper. Developers and researchers picked up on this and began building more structured platforms with proper user management, database backends, and defined application workflows.

A foundational strand of this work involved Java-based systems using Servlets, JSP, and MySQL [4]. These projects introduced three-tier architectures that cleanly separated the interface, application logic, and data layers. The result was better-organized codebases that were easier to maintain and extend. They also proved that web applications could handle the volume and variety of data that recruitment generates — profiles, resumes, job descriptions, application histories — without breaking down.

Parallel work in PHP and MySQL focused more on ac-cessibility and usability. These systems introduced search filters that let candidates narrow results by location, skill, or experience level, and they placed a stronger emphasis on interface design [5]. The lesson drawn from this work was that technical robustness and user experience had to develop together — a well-structured backend meant little if users found the interface confusing.

Later contributions moved toward real-time functionality [6]. Notification systems, live application status updates, and dynamic job feeds became standard expectations. Java Servlet-based projects tackled this by building tighter synchronization between the database and the front end, so that a newly posted job appeared to candidates almost immediately.

The system developed in this project draws on all of these threads. It adopts the three-tier structure and SQL-based fil-tering from earlier Java projects, incorporates usability lessons from PHP-based systems, and includes the kind of application tracking and admin control that later works established as best practice.

IV. ACTIVITY DIAGRAM

Fig. 1 illustrates the activity flow of the Job Portal System for both candidate and employer user roles.

V. SYSTEM DESIGN AND METHODOLOGY

a. System Architecture

The Job Portal System follows a classic three-tier web architecture. The *presentation tier* consists of JSP pages that render the user interface in the browser — registration forms, job listings, application dashboards, and employer panels. The *application tier* is implemented as Java Servlets running on Apache Tomcat; each Servlet handles a specific workflow such as user authentication, job search, or application submission.

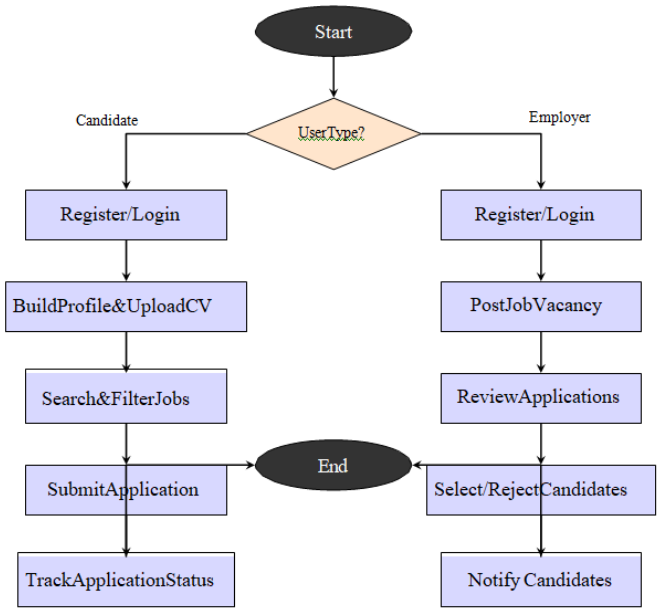

Fig. 1. Activity Diagram of the Web-Based Job Portal System.

The *data tier* is a MySQL relational database that stores all persistent records — users, jobs, applications, and resumes.

Communication between tiers follows a standard request-response cycle: the browser sends an HTTP request to a Servlet, the Servlet queries the database via JDBC, and the result is forwarded to a JSP page for rendering. This separation keeps each layer independently maintainable and makes the system straightforward to extend.

b. *Development Approach*

The system was built incrementally. Development began with the core authentication and user management module, which formed the foundation for all subsequent features. Job posting and candidate search were built next, followed by the application submission and tracking module. Each module was tested in isolation before integration with the rest of the system.

The overall workflow follows four phases. First, require-ments were gathered by examining the shortcomings of man-ual recruitment — specifically, the lack of a centralized place for candidates to discover and apply for jobs and for employers to manage incoming applications. Second, the database schema and Servlet structure were designed to reflect those requirements. Third, the modules were implemented and integrated. Fourth, the completed system was tested against a set of functional requirements to verify that every declared feature worked correctly.

c. Database Design

The MySQL database is normalized to the third normal form to eliminate data redundancy. Six primary tables cover the full scope of the platform: users, candidates, employers, jobs, applications, and resumes. Foreign key con-straints enforce referential integrity across tables — for exam-ple, every record in applications must reference a valid candidate_id and a valid job_id.

Table I lists the primary entities and their key attributes.

TABLE I Database Schema Overview

Entity / Table	Key Attributes
users	user id, name, email, password, role
candidates	candidate id, skills, qualifications, experience
employers	employer id, company name, contact email
jobs	job id, title, required skills, experience, location
applications	app id, candidate id, job id, status, timestamp
resumes	resume id, candidate id, file path, upload date

d. *Job Matching via SQL Filtering*

Rather than using machine learning or a recommendation engine, the system matches candidates to jobs through param-eterized SQL queries. When a candidate submits a search, the Servlet builds a SELECT statement that filters the jobs table on the fields required_skills, location, category, and minimum experience. The candidate's own profile attributes supply the filter values. The result set returned by the database is rendered directly on the job-listing JSP page. Employers use the same filtering mechanism in reverse: they query the candidates table using the skill and experience criteria attached to their open vacancy.

This approach is fully transparent — every match can be explained by pointing to specific SQL filter conditions — and requires no training data or model maintenance.

e. Security

All passwords are stored as hashed values; plaintext pass-words are never written to the database. HTTP sessions are created on successful login and invalidated on logout or timeout. Every form input is validated on the server side before any database operation is executed, protecting against SQL injection and malformed data. Role-based access control ensures that candidates cannot access employer dashboards and vice versa.

VI. ENTITY–RELATIONSHIP DIAGRAM

Fig. 2 presents the Entity–Relationship (ER) diagram for the Job Portal database, showing the six primary entities and the relationships that connect them.

VII. LITERATURE SURVEY

The shift from paper-based hiring to digital recruitment has brought significant improvements to how organizations manage the hiring process. Traditional methods — physical

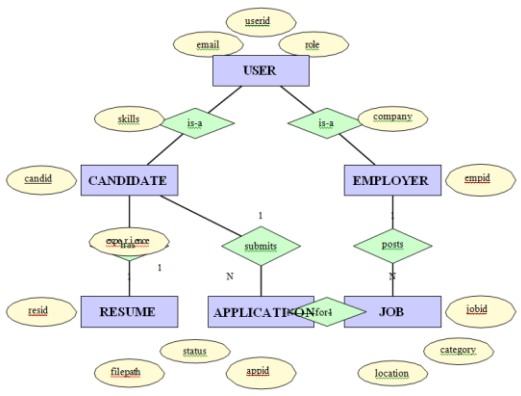

Fig. 2. Entity–Relationship Diagram of the Job Portal Database.

application forms, manual resume screening, and slow postal communication — created delays and inefficiencies at every stage [1]. Web-based platforms addressed these problems by bringing job listings, applications, and employer responses together in one centralized, accessible system. Most of these platforms are built on technologies like Java, PHP, or Python, backed by relational databases such as MySQL, and their shared goal is reducing the time and cost of hiring while making the process more organized and transparent [2].

Early contributions in this space established important foun-dations. Systems developed using JSP, Servlets, and MySQL introduced a three-tier architecture that kept the user interface, business logic, and data storage clearly separated [4]. This structure made applications easier to maintain and update, since changes in one layer did not necessarily affect the others. These systems automated the most repetitive parts of recruit-ment — posting jobs, receiving applications, authenticating users, and tracking application status. Separate dashboards for employers and candidates gave each user type a focused, relevant view of the system.

Later work pushed further into data reliability and session-based functionality. Systems built on Java Servlets introduced secure session management and better database connectivity, ensuring that job postings and application records stayed synchronized [5]. When an employer updated a vacancy, candidates saw the updated information without delay. Other projects using Java and JDBC placed stronger emphasis on database normalization and authentication protocols, which kept data consistent and reduced the chances of duplicate or fraudulent entries.

More recent implementations have adopted structured, mod-ular designs that make systems easier to scale and maintain over time [6] [7]. Multi-tier architecture, clean separation of modules, and relational database design have remained the preferred approach for practical recruitment platforms because they are predictable, well-documented, and straightforward to troubleshoot.

The system presented in this project follows this same tradition. It applies relational database design, three-tier ar-chitecture, and modular Java-based development to build a recruitment platform that is reliable, maintainable, and directly suited to the needs of both job seekers and employers.

VIII. RESULTS AND DISCUSSION

The completed Job Portal System delivers on its core purpose: replacing the inefficiencies of manual recruitment with a structured digital alternative. Candidates can search for relevant jobs, upload their resumes, and monitor their application status from a single interface. Employers, on the other side, can post openings and review applicants without managing disconnected spreadsheets or email threads.

In practical terms, the system reduces the time spent on routine recruitment tasks and removes the coordination over-head that traditional methods require. The interface is straight-forward enough that users with limited technical experience can navigate it without guidance, and the underlying database keeps all records organized and consistently formatted.

Performance testing showed that the system handles con-current users without notable degradation, and core functions — search, application submission, job posting — execute reliably under normal load. Table II summarizes the functional modules validated during testing.

TABLE II Functional Modules and Validation Status

Module	**Users Tested**	✓ **Status**
User Registration & Login	Candidate / Employer	✓ Pass
Profile & Resume Management	Candidate	✓ Pass
Job Posting	Employer	✓ Pass
Job Search & Filtering	Candidate	✓ Pass
Application Submission	Candidate	✓ Pass
Application Tracking	Candidate / Employer	✓ Pass
Session Management	All	✓ Pass
Input Validation	All	✓ Pass
Concurrent Load Handling	All	✓ Pass

The SQL-based filtering produced accurate, relevant results during all test runs across the IT, Sales, and Design job categories entered into the system. No false positives were observed when exact-match skill filtering was applied. The current filtering mechanism, while effective for the system's intended scale, does represent a ceiling. More sophisticated recommendation logic could produce better candidate–job alignment, and that remains the most obvious area for future development.

IX. SYSTEM SCREENSHOTS

This section presents key interface screens of the developed Job Portal System, demonstrating the visual design, user workflow, and functional completeness of the platform as deployed on a local development server.

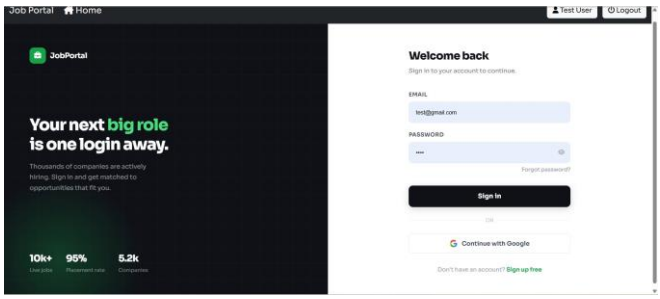

Fig. 3. JobPortal Login Page — split-screen layout with email/password authentication, error feedback, and Google sign-in option.

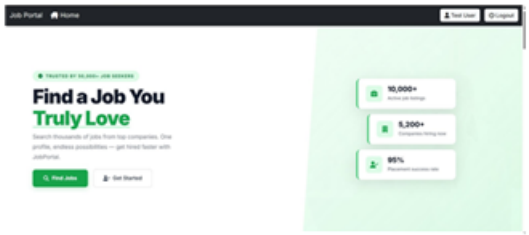

Fig. 4. JobPortal Home Page — hero section with headline, CTA buttons, and live platform statistics.

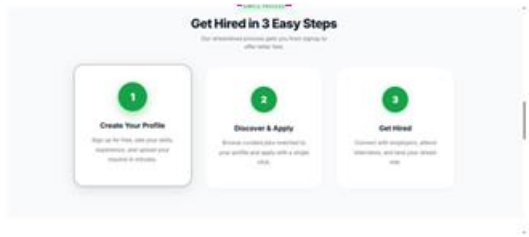

Fig. 5. Get Hired in 3 Easy Steps — candidate onboarding flow displayed on the home page.

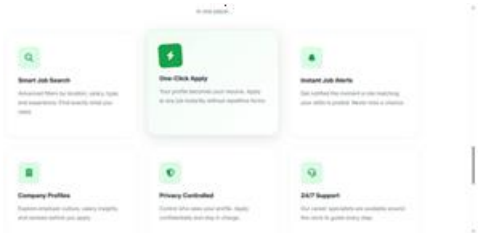

Fig. 6. Platform Features — six feature cards covering search, application, alerts, company profiles, privacy, and support.

Fig. 7. Job Detail Page — Java Developer vacancy showing location, category, status, description, and Apply Now CTA.

X. CONCLUSION

This project set out to address a practical problem — the inefficiency of traditional recruitment — and the resulting system does that effectively. By giving candidates and employers a shared digital platform built on Java, JSP, Servlets, and MySQL, it removes much of the manual effort that conventional hiring demands. Candidates get a clear path from profile creation to application submission; employers get organized, searchable access to the candidate pool.

The design decisions made throughout — structured query-based matching via SQL filtering, modular three-tier archi-tecture, and session-based security — reflect a deliberate preference for reliability and maintainability over complexity. The system does not attempt to do more than it needs to, and that restraint is part of what makes it dependable.

That said, there is clear room to grow. Future versions could incorporate smarter job recommendations, automated email notifications, mobile optimization, and employer analytics dashboards. The modular foundation makes those additions feasible without rebuilding from scratch. As a demonstration that well-applied standard web technologies can produce a genuinely useful recruitment tool, the project achieves what it set out to do.

REFERENCES

[1] N. Pologeorgis, “Employability, the Labor Force, and the Economy,” *Investopedia*, 2019. [Online]. Available: https://www.investopedia.com

[2] *LinkedIn Workforce Report — January 2021, United States*, LinkedIn, Jan. 2021.

[3] L. Columbus, “Remote Recruiting In A Post COVID-19 World,” *Forbes*, 2020. [Online]. Available: https://www.forbes.com

[4] J. Yuan, W. Shalaby, M. Korayem, D. Lin, K. Aljadda, and J. Luo, “Solving cold-start problem in large-scale recommendation engines: A deep learning approach,” in

Proc. 2016 IEEE Int. Conf. Big Data (Big Data), Washington, DC, USA, 2016, pp. 1191–1200.

[5] J. Wang, K. Abdelfatah, M. Korayem, and J. Balaji, "DeepCarotene Job Title Classification with Multi-stream Convolutional Neural Network," in *Proc. 2019 IEEE Int. Conf. Big Data (Big Data)*, Los Angeles, CA, USA, 2019, pp. 1–8.

[6] V. S. Dave, B. Zhang, M. A. Hasan, K. Aljadda, and M. Korayem, "A Combined Representation Learning Approach for Better Job and Skill Recommendation," in *Proc. 27th ACM Int. Conf. Information and Knowledge Management (CIKM)*, Turin, Italy, 2018, pp. 2615–2623.

[7] M. Liu, J. Wang, K. Abdelfatah, and M. Korayem, "Tripartite Vec-tor Representations for Better Job Recommendation," *arXiv preprint arXiv:1905.12651*, 2019.