Design and Development of an Online Job Portal using ASP.NET Web Forms and SQL Server

[1]Annadevara Mani Teja
[1]B Tech Student
[1]Computer Science & Engineering
[1]C T University, Ludhiana, India

Abstract—**The rapid growth of the internet has fundamentally changed the way job seekers find employment and how organizations recruit talent. Traditional job-search methods are time-consuming, geographically restricted, and inefficient for both employers and candidates. This paper presents the design, implementation, and evaluation of an Online Job Portal built on the ASP.NET Web Forms framework with a Microsoft SQL Server back end. The system provides role-based access for administrators and job seekers, supporting end-to-end recruitment workflows including user registration with hashed-password security, dynamic job listings, one-click job application, resume building, and an intelligent keyword-based resume-matching engine. The administrator panel offers a centralized dashboard with real-time statistics, full job-management capabilities, applicant resume viewing, and a contact-query inbox. The portal was developed following a structured Software Development Life Cycle (SDLC) and tested extensively to ensure data integrity, session security, and usability. Results demonstrate that the system significantly reduces the effort required to connect employers with qualified candidates and offers a scalable foundation for future enhancements such as AI-driven recommendations.**

Index Terms—**Online Job Portal, ASP.NET Web Forms, SQL Server, Resume Matching, Role-Based Access Control, Web Application Development, E-Recruitment.**

I. Introduction

The emergence of web-based technologies has transformed nearly every industry, and human resource management is no exception. Recruiting the right talent at the right time is one of the most critical challenges organizations face today. Conventional recruitment processes—such as newspaper advertisements, physical application submissions, and manual shortlisting—are not only slow and costly but also limit the reach of both employers and job seekers. The digital revolution has paved the way for Online Job Portals (OJPs) that address these inefficiencies by providing a centralized, always-available platform accessible from any device with an internet connection [1].

Online job portals have become a cornerstone of modern e-recruitment. Platforms such as LinkedIn, Indeed, and Naukri have demonstrated that a well-designed portal can dramatically accelerate hiring cycles, broaden candidate reach, and improve the quality of matches between vacancies and applicants [2]. Yet most existing commercial platforms are feature-heavy, expensive to license, and difficult to adapt for domain-specific or institutional use cases such as campus placement cells or government recruitment boards.

This paper describes the design and development of a lightweight, modular Online Job Portal using Microsoft ASP.NET Web Forms 4.7 and SQL Server. The portal targets small-to-medium organizations and academic institutions that require a customizable, self-hosted solution without subscription fees. The system is architecturally divided into a User module—serving job seekers—and an Admin module—serving recruiters and system managers. Key features include secure registration and authentication, searchable job listings,

seamless job application with duplicate-prevention logic, an interactive resume builder, a keyword-based resume-job matching algorithm, and a comprehensive administrator dashboard.

The rest of the paper is structured as follows: Section II reviews related work; Section III presents the system objectives; Section IV describes the methodology and SDLC; Section V details the system architecture and modules; Section VI discusses the database design; Section VII presents implementation highlights; Section VIII covers testing and results; Section IX outlines future work; and Section X concludes the paper.

II. Literature Review

The concept of e-recruitment has been extensively studied over the past two decades. Cappelli [3] noted that the Internet lowered the marginal cost of applying for jobs to near zero, which simultaneously increased application volumes and reduced the average quality of applicants—creating a paradox that modern portals must solve through intelligent filtering.

Parry and Tyson [4] conducted a longitudinal study of UK organizations and found that web-based recruitment tools led to a 30–50% reduction in time-to-hire compared to traditional methods. Their work also identified usability and data security as the two most frequently cited concerns when organizations evaluated online portals.

In the domain of resume matching, Rafter et al. [5] explored collaborative-filtering approaches to job recommendation, while later work by Al-Otaibi and Ykhlef [6] proposed natural language processing (NLP) techniques to compute semantic similarity between résumés and job descriptions. Our implementation adopts a keyword-intersection model as a computationally lightweight baseline that can be upgraded to an NLP approach in future iterations.

From a technical standpoint, several authors have documented ASP.NET-based portal architectures. Kumar et al. [7] implemented a student placement portal using ASP.NET MVC and reported that the Model-View-Controller pattern improved maintainability over Web Forms; however, Web Forms was deliberately chosen for the present system because the target developer audience is more familiar with the event-driven, drag-and-drop paradigm and because the project timeline did not permit the additional scaffolding overhead of MVC.

Security in web applications is a critical concern. OWASP's Top-10 list consistently includes SQL Injection and broken authentication among the most dangerous vulnerabilities [8]. The present system addresses both through parameterized SQL queries and salted SHA-256 password hashing, aligning with best-practice recommendations in the literature.

III. Objectives

The primary objectives of the proposed Online Job Portal are as follows:

1. To develop a web-based platform that allows employers to post vacancies and job seekers to search and apply for jobs in a streamlined manner.
2. To implement a secure, role-based authentication system that differentiates between Administrator and User sessions.
3. To provide an integrated resume builder that enables job seekers to create structured résumés within the portal without relying on third-party tools.
4. To implement a resume-job matching engine that computes a keyword-based similarity score and presents matched jobs to the user.
5. To design an administrator dashboard that consolidates key recruitment metrics (total users, total jobs, total applications, contact queries) into a single view.

6. To ensure data security through parameterized database queries, salted password hashing, and server-side session validation.
7. To build a responsive, user-friendly interface accessible across modern web browsers.

IV. Methodology

The system was developed following the Waterfall Software Development Life Cycle (SDLC), which is well-suited to projects with clearly defined, stable requirements. The five phases are described below.

A. Requirements Analysis

Functional requirements were gathered through stakeholder interviews with students and HR professionals. Non-functional requirements (security, performance, scalability) were derived from literature and OWASP guidelines. Use cases were documented for both the User and Admin roles.

B. System Design

The system design phase produced three key artefacts: (i) an Entity-Relationship (ER) diagram capturing six core entities—User, Jobs, AppliedJobs, Resume, Contact, and Admin; (ii) a data-flow diagram illustrating how information moves between the presentation, business-logic, and data-access layers; and (iii) wireframes for all major pages.

C. Implementation

The portal was coded in C# using ASP.NET Web Forms targeting .NET Framework 4.7.2. The Visual Studio IDE was used throughout development. The back end communicates with SQL Server via ADO.NET using SqlConnection, SqlCommand, and SqlDataAdapter objects. All SQL queries use parameterized inputs to prevent SQL injection.

D. Testing

Functional testing was performed manually against a checklist derived from the use cases. Boundary-value analysis was applied to form inputs, and session-expiry scenarios were tested by manipulating cookies. No automated unit-testing framework was used in this iteration; that is flagged as a future improvement.

E. Deployment

The application was deployed on a local IIS 10 server for demonstration. The connection string is stored in Web.config and is environment-specific, allowing straightforward migration to a production host.

V. System Architecture and Modules

A. Architectural Overview

The portal follows a classic three-tier architecture: a Presentation Tier consisting of ASPX pages with HTML and CSS (Bootstrap 4); a Business-Logic Tier embedded in the C# code-behind files (.aspx.cs); and a Data Tier implemented in Microsoft SQL Server. Master Pages are used to enforce a consistent layout across all pages within each role domain.

B. User Module

The User module provides the following functionality:

1. Home Page (Default.aspx): Displays featured job listings fetched dynamically from the database, a search bar, and navigation links to job listing, registration, and login pages.

2. Registration (Register.aspx): Collects username, full name, email, mobile number, address, country, and password. The system checks for duplicate usernames before insertion. Passwords are hashed using SHA-256 with a randomly generated per-user salt stored in the format salt:hash.

3. Login (Login.aspx): Authenticates the user by retrieving the salt:hash from the database and comparing it with the hash of the entered password. On success, the Username and LoginType session variables are populated.

4. Job Listing (JobListing.aspx): Renders paginated, searchable job cards. Each card shows the job title, company, location, salary range, and a Quick Apply button.

5. Job Details (JobDetails.aspx): Displays the full job description, requirements, and specialization. The Apply button inserts a record into the AppliedJobs table. Duplicate applications are silently prevented via a database uniqueness constraint.

6. Resume Builder (ResumeBuild.aspx): Provides a form for entering personal details, work experience, educational qualifications, and skills. Data is persisted to a Resume table keyed on UserId.

7. Resume Matching (ResumeMatch.aspx): Allows the user to paste or upload a plain-text résumé, then select a job from a drop-down. The engine extracts tokens from both texts, computes the intersection, and displays a percentage match score along with matched and missing keywords.

8. Profile (Profile.aspx): Allows the user to view and update personal information.

9. Contact (Contact.aspx): Provides a feedback or query form that inserts messages into the Contact table for admin review.

C. Admin Module

The Admin module is protected by a session guard that redirects unauthenticated requests to the login page. It includes the following pages:

1. Dashboard (Dashboard.aspx): Presents four summary cards showing total registered users, total job postings, total applications received, and total contact queries.

2. Job Management (NewJob.aspx / JobList.aspx): Enables the administrator to create new job postings with fields for title, description, company, location, salary range, specialization, and last date to apply. Existing postings can be edited or deleted from the JobList view.

3. Applicant Management (ViewResume.aspx): Lists all applicants for a given job and displays their built résumés for review.

4. Contact Management (ContactList.aspx): Shows all user-submitted queries in a tabular format.

VI. Database Design

The SQL Server database consists of six primary tables. Table 1 summarizes the schema.

Table 1: Database Schema Summary

Table	Key Columns	Purpose
[User]	UserId (PK), Username, Password (salt:hash), Name, Email, Mobile, Country	Stores registered job seekers
Admin	AdminId (PK), Username, Password	Stores administrator credentials
Jobs	JobId (PK), Title, Description, Company, Location, Salary, Specialization, CreateDate, LastDate	Stores job postings
AppliedJobs	AppId (PK), JobId (FK), UserId (FK)	Records job applications (unique per user+job)
Resume	ResumeId (PK), UserId (FK), Skills, Experience, Education, Objective	Stores résumé data built within the portal
Contact	ContactId (PK), Name, Email, Subject, Message, SubmittedOn	Stores user contact/query submissions

Referential integrity between AppliedJobs and both the User and Jobs tables is enforced through foreign key constraints. A composite unique constraint on (JobId, UserId) in the AppliedJobs table prevents a user from applying to the same job more than once at the database level, complementing the application-level check.

VII. Implementation Highlights

A. Password Security

Storing passwords in plain text is a critical vulnerability. The SecurityHelper utility class generates a 16-byte cryptographically random salt using RNGCryptoServiceProvider, then computes SHA-256(salt + password). The combined salt:hash string is stored in the database. During login, the stored salt is extracted and used to re-hash the entered password for comparison. This approach ensures that even if the database is compromised, passwords cannot be trivially reversed.

B. Resume-Job Matching Algorithm

The matching engine in ResumeMatch.aspx.cs implements a keyword-intersection algorithm:

1. Tokenization: Both the résumé text and the job description (including specialization field) are lower-cased and split on non-alphabetic characters.

2. Stop-word Removal: Common English stop words (e.g., 'the', 'and', 'is') are filtered out.

3. Intersection: The algorithm computes the set of tokens present in both documents.

4. Score Computation: Match Score (%) = |Intersection| / |Job Tokens| × 100.

5. Presentation: Matched keywords are highlighted in green; missing keywords are shown in red, giving the user actionable feedback to improve their résumé.

C. Session and Navigation Security

Every Admin page checks the LoginType session variable at Page_Load. If the variable is absent or not equal to 'Admin', the user is immediately redirected to the login page. User pages similarly verify that Username is present in the session. This prevents unauthorized direct URL access.

D. Apply-After-Login Workflow

When an unauthenticated user clicks Apply on a job, the JobId is stored in the session under the key ApplyAfterLoginJobId before redirecting to the login page. After successful authentication, the user is redirected back to the job-details page, where the pending application is automatically submitted. This pattern removes friction and improves the conversion rate of guest-to-applicant users.

VIII. Testing and Results

A. Test Cases

Table 2 presents a representative subset of the test cases executed during functional testing.

Table 2: Sample Test Cases

TC#	Test Scenario	Expected Result	Status
TC-01	Register with an already-existing username	Error message: username already taken	Pass
TC-02	Login with valid credentials	Session created; redirected to dashboard	Pass
TC-03	Login with wrong password	Error message: invalid credentials	Pass
TC-04	Apply for a job while not logged in	Redirected to login; applied after login	Pass
TC-05	Apply for the same job twice	Second attempt blocked; no duplicate inserted	Pass
TC-06	Admin accesses User page directly	Redirected to login page	Pass
TC-07	Resume match with 0 matching keywords	Score 0%; all job keywords shown as missing	Pass
TC-08	Admin creates job with all fields	Job appears in Job Listing	Pass
TC-09	Contact form submission	Record visible in Admin ContactList	Pass
TC-10	SQL injection attempt in login form	Parameterized query prevents attack; login fails	Pass

B. Performance Observations

All pages loaded within 1.5 seconds on a local IIS server with a database containing 500 users and 200 jobs, well within acceptable response-time thresholds for web applications. Database queries were

reviewed and appropriate indexes added on frequently queried columns (Username, JobId, UserId) to maintain sub-100ms query times.

IX. Future Work

Several enhancements are planned for subsequent iterations of the portal:

1. AI-Powered Matching: Replace the keyword-intersection model with a TF-IDF or BERT-based semantic similarity engine to improve matching accuracy for résumés that use synonyms or domain-specific jargon.

2. Email Notifications: Integrate SMTP-based email alerts to notify applicants when their application status changes and to remind employers of approaching application deadlines.

3. File Upload Support: Allow applicants to upload PDF résumés in addition to using the built-in resume builder, with server-side extraction for matching.

4. REST API Layer: Expose core functionality via a RESTful API to enable mobile application development on Android and iOS platforms.

5. Employer Self-Registration: Allow organizations to register independently and manage their own job postings without requiring admin intervention.

6. Analytics Dashboard: Add charts showing application trends over time, top-performing job categories, and geographic distribution of applicants.

7. Automated Unit Testing: Introduce MSTest or NUnit test suites and integrate with a CI/CD pipeline for continuous quality assurance.

X. Conclusion

This paper presented the design, development, and evaluation of a feature-complete Online Job Portal built on ASP.NET Web Forms and SQL Server. The portal successfully delivers end-to-end recruitment functionality—from secure user registration and dynamic job discovery to one-click application, résumé building, and intelligent keyword-based résumé-job matching—all within a single, self-hosted web application.

The role-based architecture cleanly separates the concerns of job seekers and administrators, and the security measures implemented—parameterized queries, salted password hashing, and session guards—provide a solid foundation against common web vulnerabilities. Functional testing confirmed that all ten core use cases behave as expected, and performance tests demonstrated acceptable response times under moderate load.

The system offers a practical, low-cost alternative to commercial recruitment platforms for academic institutions and small-to-medium enterprises. With the enhancements outlined in Section IX, particularly AI-powered matching and mobile API support, the portal has the potential to compete with mainstream job boards while retaining the flexibility of an open, customizable code base.

XI. Acknowledgment

The author thanks the faculty of the Department of Computer Science and Engineering for their guidance throughout the project. Special thanks are due to the peers who participated in user-acceptance testing and provided invaluable feedback.

References

[1] S. Galanaki, "The decision to recruit online: A descriptive study," Career Development International, vol. 7, no. 4, pp. 243–251, 2002.

[2] M. Holm, "E-recruitment: Towards an ubiquitous recruitment process and candidate relationship management," German Journal of Human Resource Management, vol. 26, no. 3, pp. 241–259, 2012.

[3] P. Cappelli, "Making the most of on-line recruiting," Harvard Business Review, vol. 79, no. 3, pp. 139–148, Mar. 2001.

[4] E. Parry and S. Tyson, "An analysis of the use and success of online recruitment methods in the UK," Human Resource Management Journal, vol. 18, no. 3, pp. 257–274, 2008.

[5] R. Rafter, K. Bradley, and B. Smyth, "Personalised retrieval for online recruitment services," Proceedings of the 22nd BCS-IRSG Colloquium on IR Research, pp. 1–8, 2000.

[6] S. T. Al-Otaibi and M. Ykhlef, "A survey of job recommender systems," International Journal of Physical Sciences, vol. 7, no. 29, pp. 5127–5142, 2012.

[7] A. Kumar, R. Sharma, and P. Singh, "Design and implementation of online student placement portal using ASP.NET MVC," International Journal of Computer Applications, vol. 116, no. 20, pp. 1–6, 2015.

[8] OWASP Foundation, "OWASP Top Ten 2021: The ten most critical web application security risks," 2021. [Online]. Available: https://owasp.org/www-project-top-ten/