**Project Report: AI Based Resume Screening & Job Matching System**

**by – Mayank Raj & Rishabh Raj**

**Project Title**

**AI-Based Resume Screening & Job Matching System**

**Problem Statement**

Modern recruitment faces several major challenges:

* **Overwhelming Resume Volume:** HR teams receive thousands of resumes per job post, making manual screening inefficient.
* **Poor Job-Candidate Matching:** Candidates often apply for jobs that don't truly match their skills or experience.
* **Bias in Hiring:** Manual screening can introduce unconscious bias, leading to unfair selection processes.

**Objective**

The objective of this project is to create a secure and efficient web-based portal that allows two types of users—**Admins** and **Clients**—to interact with the system in different ways:

* **Admins** authenticate using traditional username/password and access the **Applicant Board**.
* **Clients** authenticate via **Google OAuth** and are directed to a **Job Application Form** whose responses are stored in a **MongoDB** database for admin review and **server.js** that run the server.

**System Overview**

This project is a full-stack application built using **Node.js**, **Express.js**, **MongoDB**, **HTML**, and **API** for Google authentication. The system separates access logic for **Admin** and **Client** roles and integrates secure data handling via a backend server. The server is made of **server.js** to host the server.

**User Roles & Workflows**

**Admin User**

* Accesses a secure login page.
* Provides a valid **username and password**.
* Upon successful login, is redirected to the **Applicant Board**.
* Views job applications submitted by clients.
* Applicant data is fetched in real time from the MongoDB database.

**Client User**

* Clicks "Client Login" and is redirected to **Google OAuth** login.
* Upon successful Google authentication:
  + Redirected to a **Job Application Form**.
  + Fills in personal details and a message.
  + Form submission triggers server-side processing and **data is stored in MongoDB**.
  + Admins will later review this application.

**Server**

* Server used to run the website.
* Save the Data from Job Form to **MongoDB.**
* Having an **AI function** used to scoring all the applicant according to their profile.
* Sending data from MongoDB to **Applicant-Board**.

**System Architecture**

**1. Frontend (HTML/CSS)**

* Basic static HTML interface for:
  + Front page with two buttons (Admin Login, Client Login)
  + Admin login form
  + Client job application form
  + Applicant scoring Board

**2. Backend (Node.js + Express.js)**

* Handles routing, data processing, error management, and user authentication.
* Divided into routes:
  + /admin routes handle login and dashboard
  + /client routes handle Google authentication and form submission

**3. Database (MongoDB)**

* Stores client-submitted job applications.
* Uses **Mongoose** for schema definition and data modeling.

**4. Server(server.js)**

* Server used to run the website.
* Save the Data from Job Form to **MongoDB.**
* Having an **AI function** used to scoring all the applicant according to their profile.
* Sending data from MongoDB to **Applicant-Board**.

**5. Authentication**

* **Admin**: Simple server-side validation with pre-defined credentials.
* **Client**: Uses **API with Google OAuth 2.0** for secure third-party login.

**Key Technologies Used**

| **Component** | **Technology** |
| --- | --- |
| Server | Node.js, Express.js |
| Authentication | Google OAuth 2.0 |
| Database | MongoDB with Mongoose |
| Frontend | HTML (static) |
| Environment | (.env file) |
| Session Handling | express-session |

**Functional Modules**

**1. Landing Page**

* Entry point of the system.
* Provides two buttons:
  + **Admin Login**
  + **Client Login via Google**

**2. Admin Login Page**

* Accepts admin credentials.
* Verifies against predefined values or stored data.
* Upon success, redirects to **Applicant Dashboard**.

**3. Applicant Board (Admin Dashboard)**

* Displays all stored applications.
* Pulls data from MongoDB.
* Organized format for review (e.g., Name, Email, Message, Date).

**4. Google Authentication (Client Side)**

* Secure login using Google accounts.
* Redirects authenticated users to the job form.

**5. Job Application Form**

* Inputs: Name, Email, Message.
* Data is sent via POST request.
* Backend saves the form data to MongoDB.

**6. Error Handling**

* Centralized error catching in the server.
* Logs or returns friendly error messages to the frontend.

**7. Server**

* Having an **AI function** used to scoring all the applicant according to their profile.
* Sending data from MongoDB to **Applicant-Board**.

**8. Module**

* It includes file having structure of JSON format to save data in MongoDB
* Use the JS to link between the frontend js and sever js.

**Security Considerations**

* Admin access protected via login credentials.
* Client access validated using secure Google OAuth.
* MongoDB communication handled securely with connection strings stored in .env.
* Sessions managed using express-session.

**Advantages of the System**

* **Secure Access** for both Admin and Clients.
* **Modular Structure**: Easy to scale or modify.
* **Real-Time Access** to data for Admins.
* **Professional Client Experience** through Google sign-in.
* **Faster Hiring Process** Reduces resume screening time by up to 80%.
* **Better Job Matches** Ensures candidates are matched based on relevant skills, not just keywords.
* **Bias Mitigation** Objective screening removes human bias from the shortlisting phase.

**Limitations & Future Enhancements**

| **Limitation** | **Potential Improvement** |
| --- | --- |
| Static Admin credentials | Use hashed passwords with a user DB |
| Basic HTML interface | Upgrade UI with React or a CSS framework |
| No email confirmation | Add email notifications for submissions |
| No application tracking | Implement status updates for applicants |
| Real-time chat with recruiter | Integrate AI chatbot for candidate queries | |

|  |  |
| --- | --- |
| Resume builder tool | Help applicants tailor resumes to jobs |

|  |  |
| --- | --- |
| Dashboard for analytics | Visualize match stats, hiring rates, etc. |

|  |  |
| --- | --- |
| Role-based access | Admin, Recruiter, Candidate portals |

**Conclusion**

This job portal system fulfills its primary goal of offering **dual-access functionality**—secure login for **Admins** and **Clients**—along with **data storage and management** capabilities. Its modular design and use of popular web technologies make it scalable, secure, and ready for production-level enhancement.