# Job Portal with Resume Matching

# 1. Introduction

This project is a Flask-based job portal where HR can upload job descriptions, and users can view and apply for jobs. The system includes a resume parser that extracts skills from resumes and matches them with job descriptions. If at least 50% of the required skills match, the resume is forwarded to HR; otherwise, the applicant is disqualified.

# 2. Objectives

- Allow HR to upload, view, and manage job postings.
- Enable users to browse and apply for jobs.
- Implement resume parsing for extracting skills.
- Match extracted skills with job requirements and determine qualification.
- Notify HR of qualified candidates and inform users of their application status.

# 3. Technologies Used

- **Backend**: Python, Flask, SQLite3
- **Frontend**: HTML, CSS, Bootstrap
- **Resume Parsing**: Python libraries (e.g., PyPDF2, pdfminer, or docx2txt)
- **Natural Language Processing (NLP)**: NLTK or scikit-learn for skill extraction
- **Database**: SQLite3 for storing job postings and applications

# 4. System Architecture

## 4.1 User Roles

- **HR (Admin)**: Uploads job descriptions, views applications.
- **User (Job Seeker)**: Views available jobs, applies by uploading resumes.

## 4.2 Workflow

1. HR uploads a job description, specifying required skills.
2. Users browse job listings and apply by uploading their resumes.
3. The resume parser extracts skills from the user's resume.
4. The system calculates the percentage match with the job description.
5. If the match is 50% or more, the application is forwarded to HR.
6. If the match is below 50%, the user is notified of disqualification.

# 5. Implementation Details

## 5.1 Database Schema

### Tables

- **users** (id, name, email, password, role)
- **jobs** (id, title, description, required_skills, posted_by)
- **applications** (id, user_id, job_id, resume_path, match_percentage, status)

## 5.2 API Endpoints

- `POST /hr/upload-job` → HR uploads a job
- `GET /jobs` → List available jobs
- `POST /apply-job` → User applies for a job (uploads resume)
- `GET /applications/<user_id>` → Fetch user's job applications

# 6. Resume Parsing & Skill Matching

- **Step 1**: Convert resume to text using `PyPDF2` or `docx2txt`.
- **Step 2**: Extract skills using NLP (predefined skill database + tokenization).
- **Step 3**: Compute skill match percentage with job requirements.
- **Step 4**: If `match_percentage >= 50%`, forward to HR; else, reject.

# 7. Expected Outcomes

- A functional job portal where HR can manage job postings.
- Automated resume parsing and skill matching system.
- Efficient filtering of applicants based on required skills.
- Improved job application process with instant feedback to users.

# 8. Future Enhancements

- Integrate machine learning for better skill matching.
- Implement email notifications for application status.
- Support multiple resume formats (PDF, DOCX, TXT).
- User profile creation with saved resumes and job alerts.

# 9. Conclusion

This project provides an efficient way to streamline the job application process by automating resume parsing and skill matching. It ensures that only qualified candidates are forwarded to HR, reducing manual effort and improving hiring efficiency.