Computer Engineering Department

Report of Job Recommender

Course Code: 3CP05

Course Name: Advance Programming Laboratory

Faculty Coordinator: Prof Mosin Hasan

Date: 11th November 2025

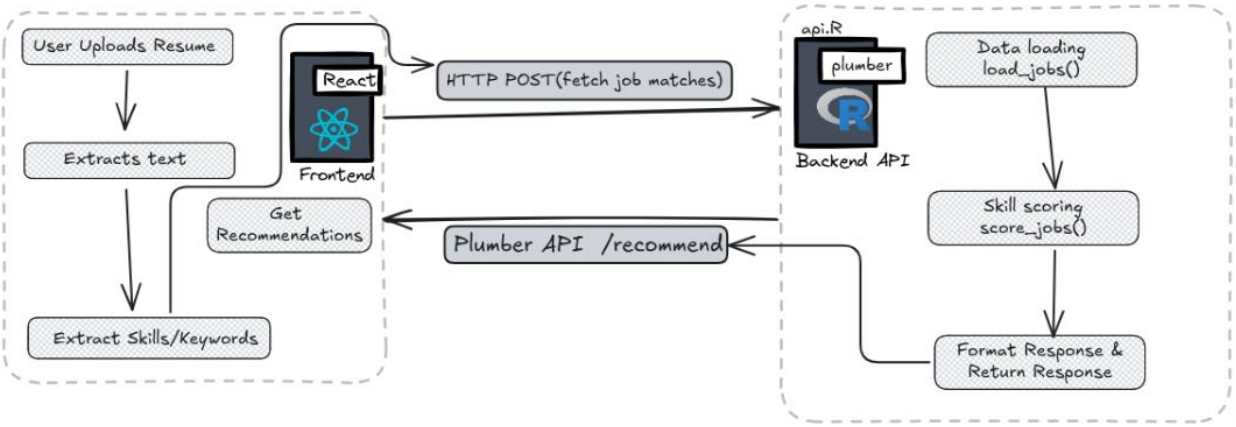
Prepared By – Mukund Parmar (23CP014)

Jaivik Prajapati (23CP002)

Overview

PDF Skill Extractor & Job Recommender is a web application that extracts text and technical skills from uploaded PDF resumes and returns personalized job recommendations using natural language processing and machine learning techniques. The app displays job recommendations with match scores, matched skills, and posting dates.

Architecture



1. **Resume Upload (Frontend - React)**
    a. Users upload their resume in PDF format through the React interface.
    b. **pdf.js** is used to extract text content from the uploaded PDF.
2. **Skill Extraction**
    a. The extracted text is matched against a predefined skills database to identify relevant technical skills.
3. **API Request to Backend**
    a. The React frontend sends a **POST** request to the **R Plumber API** endpoint /recommend, containing the extracted skills as JSON data.
4. **Job Matching Logic (Backend - R Plumber API)**
    a. The backend processes the skills and matches them against a job dataset.
    b. Calculates match scores based on skill relevance

Key Features

- Client-side PDF text extraction using pdf.js
- Automatic detection of technical skills from resume content using a comprehensive skills database
- Job recommendation engine that ranks results by skill match scores
- Responsive user interface built with React and Tailwind CSS
- R-based backend (Plumber) for data processing and recommendation logic
- Detailed match results including scores, matched skills, job metadata, and posting dates
- Intelligent text formatting for job descriptions with expand/collapse functionality

Technology Stack

Frontend

- React 19.1.1
- Vite
- Tailwind CSS
- pdf.js

Backend

- R with Plumber
- dplyr, purrr, tibble, stringr, lubridate, jsonlite

Prerequisites

- Node.js v18 or later (includes npm)
- R v4.0 or later
- Required R packages: plumber, jsonlite, dplyr, purrr, tibble, stringr, lubridate

Typical Workflow

1. Upload a PDF resume through the user interface.
2. The application extracts text and identifies skills from the predefined database.
3. Submit detected skills to obtain ranked job recommendations.
4. Review detailed matches, scores, and posting dates.

Phase 1: Frontend PDF Processing (React / Browser)

1. **User Uploads Resume (PDF)**
    a. Handled by ResumeUploader.jsx.
    b. Uses **pdfjs-dist** to extract text from all PDF pages.
2. **Skill Extraction**
    a. Extracted text is matched against predefined skills.json.
3. **Auto Trigger Recommendations**
    a. If relevant skills are found, fetchRecommendations() is automatically called.

Phase 2: API Request (React → R Backend)

- Frontend sends a **POST** request to /recommend endpoint.
- JSON payload includes extracted skills and optional metadata.

Phase 3: Backend Processing (R / Plumber API)

1. **Request Handling**
    a. api.R receives POST via **Plumber** framework.
2. **CORS Configuration**
    a. Allows requests from the React frontend.
3. **Data Loading**
    a. Uses load_jobs() to load job data from enhanced_jobs_step3.json.
4. **Skill Scoring**
    a. score_jobs() computes match scores based on:
        i. Word frequency weights (word_frequency.csv)
        ii. Text similarity across titles, descriptions, and tags
        iii. Weighted multi-factor scoring algorithm

Phase 4: Data Processing Pipeline (R Scripts)

Script	Function
Cleaining_data.R	Handles data preprocessing for job postings, including installing required R packages, loading and sampling 5,000 jobs from a CSV file, converting text to lowercase, and defining a clean_text function for text cleaning (e.g., removing noise, standardizing formats).
api.R	recommendations, including CORS handling, job data loading from JSON, skill-based scoring and ranking of jobs, and a /recommend endpoint that returns top-matched jobs with calculated scores and matched skills.
normalize_descriptions.R	Processes job descriptions to improve readability by adding line breaks, formatting bullet points and lists, identifying section headers, and inserting paragraph breaks

Phase 5: Response & Frontend Display

1. **Response**
    a. API returns top 10 job matches in JSON format.
2. **Frontend Rendering**
    a. Recommendations.jsx shows results with expandable job cards.
    b. Displays match score, matched skills, and job category.
3. **UI Feedback**
    a. Smooth transitions, loading indicators, and clear result ranking.

Tech Stack

- **Frontend:** React + Vite, PDF.js for PDF parsing
- **Backend:** R + Plumber API (Port 8000)
- **Data:** JSON + CSV (pre-processed datasets)

- **Communication:** RESTful API (JSON)
- **CORS:** Enabled for development

API

POST /recommend

- **Request body (JSON):**{ "skills": ["python", "react", "sql"] }

- **Response (JSON):** List of recommended jobs with scores, matched skills, and posted dates. Example:{

```
 "results": [
  {
   "title_clean": "Data Scientist",
   "company_name": "Tech Corp",
   "location": "San Francisco, CA",
   "posted_date": "2023-08-03 12:00:13",
   "score": 85,
   "matches": 3,
   "matched_skills": ["python", "sql", "machine learning"],
   "description": "Full job description...",
   "tags": "Data Science, python, sql",
   "job_category": "Data Science",
   "seniority_level": "Senior",
   "remote_type": "Remote",
   "employment_type": "Full-time",
   "salary_range": "$100k - $150k"
  }
 ]
}
```

Project Structure

```
Job-Recommender/
├── public/
├── src/
│   ├── components/
│   │   ├── ResumeUploader.jsx
│   │   └── Recommendations.jsx
│   ├── App.jsx
│   ├── skills.json
│   └── ...
├── server/
│   └── r_backend/
│       ├── data/
│       │   └── enhanced_jobs_step3.json
│       ├── api.R
│       └── ...
├── package.json
├── vite.config.js
└── README.md
```