Similar Candidate Recommendation

with RChilli

June 2024 | Version 1.0

Purpose Statement

This document details integrating **Similar Candidate Recommendation** use case leveraging Search & Match Engine.

Disclaimer:

Contents

Introduction

The **Similar Candidate Recommendation** feature enhances recruiter efficiency by quickly and easily identifying candidates with similar skills, experience levels, and job titles from the candidate pool.

By parsing and indexing candidate resumes, the system matches these resumes with others to generate similar candidate recommendations, significantly enhancing recruiter productivity and search accuracy.

Similar Candidate Recommendation simplifies candidate search by:

1. **Uploading Resumes**: Candidates upload their resumes directly to the system.
2. **Parsing and Indexing Resumes**: The system processes the uploaded resumes using the RChilli Search & Match API.
3. **Generating Recommendations**: The system matches and recommends candidates with similar profiles based on the parsed data.

Key Components and Flow

1. **Resume Upload and Parsing:**
    a. **Action:** Candidates submit their resumes, which are parsed & indexed using the RChilli Search & Match API.
    b. **Response:** The parsed data is indexed and stored in the database.
2. **Similarity Matching:**
    a. **Action:** The system uses the indexed data to find similar candidates based on skills, experience levels, and job titles.
    b. **Response:** A list of similar candidates is generated.
3. **Recommendation Display:**
    a. **Action:** The list of similar candidates is displayed to the recruiter.
    b. **Response:** Recruiters can review and shortlist similar candidates for further consideration.

Benefits

1. **Time Efficiency**: Drastically reduces the time required to identify suitable candidates.
2. **Enhanced Precision**: Matches candidates based on detailed and comprehensive criteria, improving search accuracy by 90%.
3. **Improved Productivity:** Streamlines the recruitment process, allowing recruiters to focus on high-value tasks.

User Stories

1. **Upload Resume**
    a. As a candidate, I want to upload my resume so that I can be considered for similar job opportunities.
2. **Parse & Index Resume**
    a. As a system, I need to parse and index the uploaded resume to extract relevant data and store it in the database.
3. **Match Similar Candidates**
    a. As a recruiter, I want to find candidates with similar skills, experience, and job titles to fill positions quickly.
4. **View Recommendations**
    a. As a recruiter, I want to view a list of recommended candidates with similar profiles for efficient shortlisting.

Example Scenario

1. **Scenario:** A recruiter named Emma is looking to fill a software engineering position and has identified a strong candidate named John.
2. **Action:** Emma uses the similar candidate recommendation feature to find more candidates like John.
3. **Response:** The system analyzes John's profile, including his skills, experience, and job titles, and generates a list of similar candidates.
4. **Outcome:** Emma reviews the recommended candidates, finds several with comparable qualifications, and quickly shortlists them for further evaluation.

User Acceptance Testing (UAT)

Conduct thorough testing to ensure the **Similar Candidate Recommendation** feature accurately identifies similar candidates and integrates seamlessly with the existing recruitment process.

UAT Scenarios

UAT Scenario 1: Resume Parsing Accuracy

- **Objective:** Verify that the system accurately parses and indexes the uploaded resume.
- **Preconditions:** Candidate has a resume file ready for upload.
- **Steps:**
  - Navigate to the resume upload page.
  - Upload a resume file.
- **Expected Result:** The resume is parsed accurately, and the data is indexed correctly.
- **Acceptance Criteria:**
  - Resume parsing completes without errors.
  - Indexed data matches the resume content.

UAT Scenario 2: Similar Candidate Matching

- **Objective:** Verify that the system accurately matches and recommends similar candidates.
- **Preconditions:** Parsed resumes are available in the database.
- **Steps:**
  - Upload a candidate's resume.
  - Trigger the similarity matching process.
- **Expected Result:** The system generates a list of similar candidates.
- **Acceptance Criteria:**
  - Similar candidates are accurately matched based on skills, experience, and job titles.
  - The list of recommended candidates is relevant and accurate.

UAT Scenario 3: Recommendation Display

- **Objective:** Verify that the recommended similar candidates are displayed correctly.
- **Preconditions: S**imilar candidates have been matched and stored in the system.
- **Steps:**
  - Navigate to the recommendation page.
  - View the list of similar candidates.
- **Expected Result:** The list of similar candidates is displayed correctly.
- **Acceptance Criteria:**
  - The recommendation list is complete and accurate.
  - Recruiters can view and shortlist candidates from the list.

UAT Scenario 4: Performance and Accuracy Testing

- **Objective:** Ensure the system performs well and provides accurate results underload.
- **Preconditions:** System has a large database of parsed resumes.
- **Steps:**
  - Perform multiple similarity matching requests simultaneously.
  - Measure the response time and accuracy of the results.
- **Expected Result:** The system handles multiple requests efficiently and provides accurate recommendations.
- **Acceptance Criteria:**
  - Response times are within acceptable limits.
  - Recommendations are accurate even under load.

Test Cases

Test Case 1: Resume Upload Test

- **Objective:** Verify that a resume can be uploaded successfully.
- **Preconditions:** Candidate has a resume file ready.
- **Steps:**
  1. Navigate to the resume upload page.
  2. Click the upload button.
  3. Select and upload a resume file.
- **Expected Result:** Resume upload completes without errors.
- **Test Data:** Various resume formats (PDF, DOCX).
- **Pass Criteria:** Resume file is uploaded and available for processing.

Test Case 2: Resume Parsing Test

- **Objective:** Verify that the uploaded resume is parsed accurately.
- **Preconditions:** Resume file has been uploaded.
- **Steps:**
  1. Upload a resume file.
  2. Check the parsed data.
- **Expected Result:** Parsed data matches the resume content.
- **Test Data:** Sample resumes with different formats and content.
- **Pass Criteria:** Parsed data is accurate and complete.

Test Case 3: Similar Candidate Matching Test

- **Objective**: Verify that the system matches similar candidates accurately.
- **Preconditions**: Parsed resumes are available in the database.
- **Steps:**
  1. Upload a candidate's resume.
  2. Trigger the similarity matching process.

- **Expected Result:** The system generates a list of similar candidates.
- **Test Data:** Resumes with varying levels of similarity.
- **Pass Criteria:** Similar candidates are accurately matched and listed.

Test Case 4: Recommendation Display Test

- **Objective:** Verify that the recommended similar candidates are displayed correctly.
- **Preconditions:** Similar candidates have been matched and stored in the system.
- **Steps:**
    1. Navigate to the recommendation page.
    2. View the list of similar candidates.
- **Expected Result:** The list of similar candidates is displayed correctly.
- **Test Data:** Various matched candidate profiles.
- **Pass Criteria:** The recommendation list is complete and accurate.

Test Case 5: Error Handling for Unsupported File Formats

- **Objective:** Verify that the system handles unsupported resume file formats correctly.
- **Preconditions:** Candidate has a resume file in an unsupported format.
- **Steps:**
    1. Attempt to upload a resume file in an unsupported format.
- **Expected Result:** The system displays an appropriate error message.
- **Test Data:** Unsupported file formats (e.g., CSV, ZIP).
- **Pass Criteria**: The system prevents unsupported file formats from being uploaded and displays an informative error message.

Test Case 6: Duplicate Resume Handling

- **Objective**: Verify that the system handles duplicate resumes appropriately.
- **Preconditions**: A resume has already been uploaded and parsed.
- **Steps**:
    1. Upload a duplicate resume file.
    2. Check if the system identifies and handles the duplicate entry.

- **Expected Result**: The system detects the duplicate and provides an appropriate message or merges the data accordingly.
- **Test Data**: Duplicate resume file.
- **Pass Criteria**: Duplicate resumes are identified, and appropriate actions are taken.

Test Case 7: Partial Data Handling

- **Objective**: Verify that the system can handle resumes with missing or partial data.
- **Preconditions**: Candidate has a resume file with missing sections (e.g., no work experience).
- **Steps**:
  a. Upload a partially filled resume.
  b. Check the parsed data for completeness and correctness.
- **Expected Result**: The system parses the available data and indexes it correctly.
- **Test Data**: Resumes with missing sections (skills, education, experience).
- **Pass Criteria**: The parsed data includes all available information without errors.

Test Case 8: Multi-language Resume Parsing

- **Objective**: Verify that the system accurately parses resumes in different languages.
- **Preconditions**: Candidate has resume files in various supported languages.
- **Steps**:
  a. Upload resumes in different languages.
  b. Check the parsed data for accuracy.
- **Expected Result**: The system accurately parses and indexes resumes in all supported languages.
- **Test Data**: Resumes in different languages (e.g., English, Spanish, French).
- **Pass Criteria**: The parsed data is accurate for each language.

Test Case 9: User Interface Testing

- **Objective**: Ensure the user interface for uploading resumes and viewing recommendations is user-friendly.
- **Preconditions**: Access to the system's UI.
- **Steps**:
    a. Navigate through the resume upload and recommendation display pages.
    b. Check for UI responsiveness and ease of use.
- **Expected Result**: The UI is intuitive, responsive, and displays information correctly.
- **Test Data**: Various resume uploads and recommendation displays.
- **Pass Criteria**: The UI is user-friendly and displays data correctly.

Test Case 10: Security and Access Control

- **Objective**: Verify that access to the system is secure and controlled.
- **Preconditions**: User accounts with different permission levels.
- **Steps**:
    a. Attempt to upload and access resumes with different user roles.
    b. Check if access is restricted based on user permissions.
- **Expected Result**: The system enforces access control and secures data appropriately.
- **Test Data**: User accounts with varying permissions.
- **Pass Criteria**: Access control is enforced correctly.

Test Case 11: Resume Update and Re-indexing

- **Objective**: Verify that the system can handle updates to existing resumes and re-index them.
- **Preconditions**: A resume has been uploaded and indexed.
- **Steps**:
    a. Update an existing resume.
    b. Check if the updated resume is re-parsed and re-indexed correctly.

- **Expected Result**: The system re-parses and re-indexes the updated resume accurately.
- **Test Data**: Updated resume file.
- **Pass Criteria**: The updated data is reflected accurately in the system.

Miscellaneous

The integration depends on your application workflow. There might be different steps involved in your integration which may vary from application to application.

This document indicates basic steps which are involved, you can review your application and add other UAT, and test cases based upon your needs.

Consider data privacy, error handling, and user feedback mechanisms to ensure a robust and user-friendly implementation.

Integration Estimation

Disclaimer: This estimation is based on a sample application with simple workflow. This may vary depending on the complexity of your application and the skill set/experience of the team involved in the integration.

1. **Development:** 20-25 hours
    a. Resume Upload: 2hr
    b. Resume Parsing using RChilli Search & Match API: 3hr
    c. Similarity Matching Algorithm: 8-10hr
    d. Recommendation Display: 4-5hr
    e. Unit Test Cases: 2-3hr
    f. Bug Fixing and Optimization: 2-3hr
2. **Testing:** 10 hours
    a. Unit Tests: 3 hours
    b. Integration Tests: 3 hours
    c. UAT: 4 hours

Appendix

Search Engine API Details:

https://docs.rchilli.com/kc/c_Search_Match_overview

Get API Key and Endpoints for Search Engine:

https://docs.rchilli.com/kc/c_RChilli_search_match_API_Endpoints

Search Engine Error Code:

https://docs.rchilli.com/kc/c_RChilli_search_match_error_code

Resume Parser API Details:

https://docs.rchilli.com/kc/c_RChilli_resume_parser_rest_API

Get API key and Endpoints for Resume Parser:

https://docs.rchilli.com/kc/c_RChilli_resume_parser_API_authentication

Postman Integration and Sample Code:

https://documenter.getpostman.com/view/6003669/Szmh1GQX?version=latest#0b486ceb-a8b6-496b-b53b-73c4adbc6987

Resume Parser Response Schema:

https://docs.rchilli.com/kc/c_RChilli_resume_parser_response_Schema

Resume Parser Error Code:

https://docs.rchilli.com/kc/c_RChilli_Resume_parser_error_code

Language Supported:

https://docs.rchilli.com/kc/c_RChilli_search_match_Response_language_support