Enhanced Search & Match

with RChilli

June 2024 | Version 1.0

Purpose Statement

This document details integrating **Enhanced Search & Match** use case leveraging Search and Match API.

Disclaimer:

Contents

Introduction

The **Enhanced Search & Match** feature improves recruiter productivity by enabling accurate candidate searches through simple search, boolean, or matching functionality. Recruiters can perform simple keyword searches, complex boolean queries, and use matching functionality to find candidates with the best fit for their job openings.

This feature enhances search results by 90%, allowing recruiters to find the most relevant candidates quickly and efficiently.

Enhanced Search & Match Capabilities**:**

- **Simple Search:** Enables recruiters to enter search queries in any language, facilitating precise candidate matching based on skills, experience, and qualifications.
- **Boolean Operators**: Provides flexibility with advanced search queries using operators like AND & OR allowing for refined candidate filtering.
- **Matching Algorithms:** Utilizes sophisticated algorithms to rank candidates according to how closely their profiles align with job requirements, ensuring optimal match accuracy.

Key Components and Flow

a. **Search Query Handling:**
    a. **Action:** Users input search criteria via natural language or boolean operators.
    b. **Description:** The system interprets and processes user queries to identify relevant candidates.
    c. **Response:** Displays search results ranked by relevance to user criteria.

b. **Boolean Operators Support:**
    a. **Action:** Allows users to combine search criteria using logical operators (AND, OR NOT).
    b. **Response:** Delivers search results that meet combined criteria specified by the user.

c. **Candidate Ranking Algorithm:**
    a. **Action:** Evaluates candidate profiles against job requirements.
    b. **Description:** Utilizes algorithms to score and rank candidates based on skills, experience, and qualifications.
    c. **Response:** Provides a ranked list of candidates, highlighting top matches for review.

d. **Real-time Search Results Update:**
    a. **Action:** Updates search results dynamically as user criteria are adjusted.
    b. **Response:** Ensures users can refine search queries effectively based on updated results.

e. **User Interface (UI) Feedback:**
    a. **Action:** Displays search results and filter options through an intuitive user interface.
    b. **Description:** Enhances user experience with clear visual representation of search outcomes.
    c. **Response:** Facilitates efficient navigation and interaction with search results and filtering mechanisms.

Benefits

1. **Efficiency:** Saves recruiters' time by quickly finding relevant candidates**.**
2. **Accuracy:** Improves the accuracy of search results by 90%.
3. **Flexibility:** Supports various search methods (simple, boolean, matching).
4. **Automation:** Automates resume parsing and indexing, reducing manual effort.

User Stories

1. **Parse and Index Resumes**
    - As a system, I need to parse, and index uploaded resumes to maintain an up-to-date candidate database.
2. **Perform Simple Search**
    - As a recruiter, I want to perform a simple keyword search to quickly find candidates with specific skills.
3. **Use Boolean Search**
    - As a recruiter, I want to use boolean expressions in my search to refine and narrow down candidate results.
4. **Matching Functionality**
    - As a recruiter, I want to use the matching functionality to automatically find candidates who best fit the job requirements.

Example Scenario

1. **Simple Search**:
    a. **Scenario**: A recruiter needs to find a candidate for a software engineer position.
    b. **Action**: The recruiter enters the search term "Java Developer".
    c. **Response**: The system processes the query and returns a list of candidates with "Java Developer" in their profiles.
    d. **Outcome**: The recruiter reviews the top candidates and selects the most suitable ones for further evaluation.
2. **Boolean Search**:
    a. **Scenario**: A recruiter is looking for a marketing manager with social media experience but not entry-level.
    b. **Action**: The recruiter enters the boolean query with required and optional skill, job profile, etc.

c. **Response**: The system processes the query and returns a list of candidates who meet these criteria.
d. **Outcome**: The recruiter reviews the candidates and selects the best fits for interviews.

3. **Match Functionality**:
    a. **Scenario**: A recruiter has a job opening for a senior project manager.
    b. **Action**: The recruiter uses the match functionality to find candidates who closely match the job description.
    c. **Response**: The system compares the job description with candidate profiles and returns the best matches.
    d. **Outcome**: The recruiter evaluates the matched candidates and schedules interviews with the top choices.

User Acceptance Testing (UAT)

The **Enhanced Search & Match** UAT plan ensures the feature meets user requirements and business goals.

UAT Scenarios

UAT Scenario 1: Accuracy of Simple Keyword Search

- **Objective**: Verify the accuracy of simple keyword search results.
- **Preconditions**: Candidate database with parsed resumes.
- **Steps**:
    a. Enter a simple keyword search (e.g., "Project Manager").
    b. Review the returned candidate profiles.
- **Expected Result**: Profiles matching the keyword are accurately displayed.
- **Acceptance Criteria**: Search results are relevant and accurate.

UAT Scenario 2: Boolean Search Functionality

- **Objective**: Validate boolean search functionality.
- **Preconditions**: Candidate database with parsed resumes.
- **Steps**:
    a. Enter a boolean search query ("Marketing as required skill AND Manager as required profile" and "Executive as optional profile").
    b. Review the returned candidate profiles.
- **Expected Result**: Profiles matching the boolean criteria are accurately displayed.
- **Acceptance Criteria**: Boolean logic is correctly applied in search results.

UAT Scenario 3: Matching Functionality Accuracy

- **Objective**: Test the matching functionality for finding relevant candidates.
- **Preconditions**: Candidate database with parsed resumes and job descriptions.

- **Steps**:
    a. Enter matching criteria (e.g., job title, skills).
    b. Review the returned candidate profiles.
- **Expected Result**: Candidates matching the criteria are accurately displayed.
- **Acceptance Criteria**: Matching functionality accurately identifies relevant candidates.

UAT Scenario 4: Performance and Load Testing

- **Objective**: Ensure the system can handle multiple search requests efficiently.
- **Preconditions**:
    a. Access to the performance testing tools to stimulate concurrent user activity
    b. Database populated with a large volume of candidate profiles
- **Steps**:
    a. Simulate multiple simultaneous search requests.
    b. Monitor system performance and response times.
- **Expected Result**:
    a. System handles the load without significant performance degradation.
    b. System maintains acceptance performance level (e.g. Response time < 1ms)
- **Acceptance Criteria**:
    a. Search response times remain within acceptable limits under load.
    b. No degradation of search and match functionality observed during peak load periods.

Test Cases

Test Case 1: Basic Search Functionality

- **Objective**: Verify that basic search functionality works as expected.
- **Preconditions**:
    a. Parsed resumes are available in the database.
    b. Access to the simple or basic search interface.
- **Steps**:
    a. Navigate to the search interface of the application.
    b. Enter a basic search query (e.g., "software engineer").
    c. Initiate the search.
- **Expected Result**: Relevant candidates matching the search query are displayed.
- **Test Data**: Simple search strings.
- **Pass Criteria**:
    a. The search results include profiles with "software engineer" in their title or description.
    b. Results are displayed in a clear and organized manner.

Test Case 2: Boolean Search Test

- **Objective**: Validate boolean search functionality.
- **Preconditions**:
    a. Parsed resumes are available in the database.
    b. Access to the advanced search interface.
- **Steps**:
    a. Enter a boolean search query - (e.g., "Java as required skill AND (Spring OR Hibernate as optional skill) ").
    b. Submit the search form.

c. Review the search results.

- **Expected Result**: Profiles matching the boolean query are accurately displayed.
- **Test Data**: Boolean search strings.
- **Pass Criteria**:
    a. The search results include candidates proficient in "Java" and either "Spring" or "Hibernate" (or both).
    b. Results are displayed in a clear and organized manner.

Test Case 3: Matching Functionality Test

- **Objective**: Validate the accuracy of the matching algorithm against a job description.
- **Preconditions**:
    a. Access to a job description and candidate profiles.
    b. Matching algorithm configured and operational.
- **Steps**:
    a. Input a job description specifying required skills and qualifications.
    b. Trigger the matching functionality to identify suitable candidates.
- **Expected Result**: Candidates are ranked based on how closely their profiles match the job description.
- **Test Data**: Matching criteria.
- **Pass Criteria**:
    a. Candidates ranked higher demonstrate stronger alignment with the job requirements.
    b. The matching algorithm considers both skills and experience effectively.

Test Case 4: Error Handling in Search Queries

- **Objective:** Verify error handling capabilities during search queries.
- **Preconditions:**
    a. Access to the search feature.

b. Attempt to execute a search query with invalid syntax or unsupported parameters.

- **Steps:**
    a. Navigate to the search interface.
    b. Enter a search query with invalid syntax or unsupported parameters.
    c. Initiate the search.
- **Expected Result:** System provides appropriate error messages and handles the invalid input gracefully.
- **Pass Criteria:**
    a. An error message is displayed indicating the nature of the error (e.g., syntax error, unsupported parameter).
    b. The system does not crash and remains responsive after handling the error.

Test Case 5: Performance Testing

- **Objective:** Assess the performance of the search and matching functionalities under load.
- **Preconditions:**
    a. Access to performance testing tools.
    b. Database populated with a significant number of candidate profiles.
- **Steps:**
    a. Simulate multiple concurrent users accessing the search and matching features.
    b. Execute a series of search queries and observe system response times.
- **Expected Result:** System maintains acceptable performance levels under peak load conditions.
- **Pass Criteria:**
    a. Search results are retrieved within a reasonable response time.
    b. System resources (e.g., CPU, memory) are utilized efficiently without degradation in performance.

Test Case 6: Advanced Matching Algorithm Accuracy

- **Objective:** Validate the accuracy of the advanced matching algorithm against a complex job description.
- **Preconditions:**
    a. Access to a detailed job description and candidate profiles.
    b. Advanced matching algorithm configured and operational.
- **Steps:**
    a. Input a detailed job description specifying required skills, qualifications, and experience.
    b. Trigger the advanced matching functionality to identify suitable candidates.
- **Expected Result:** Candidates are ranked based on how closely their profiles match the detailed job description.
- **Test Data:** Complex matching criteria including multiple skills, experience levels, and qualifications.
- **Pass Criteria:**
    a. Candidates ranked higher demonstrate stronger alignment with the job requirements.
    b. The matching algorithm considers multiple factors such as skills, experience, and education effectively.

Test Case 7: User Interface Interaction

- **Objective:** Ensure the user interface for search and match features is intuitive and responsive.
- **Preconditions:** Access to the application's search and match interface.
- **Steps:**
    a. Navigate to the search interface of the application.
    b. Perform a simple search, a boolean search, and use the matching functionality.
    c. Interact with the search results by applying filters, sorting, and selecting candidates.

- **Expected Result:** The user interface should be intuitive, responsive, and provide clear visual representation of search results.
- **Test Data:** Various search queries and interaction patterns.
- **Pass Criteria:**
    a. The search interface responds promptly to user inputs.
    b. Filters and sorting options work correctly.
    c. The layout and presentation of search results are user-friendly and informative.

Test Case 8: Edge Case Handling for Search Queries

- **Objective:** Verify the system's behavior when handling edge case search queries.
- **Preconditions:** Access to the search feature.
- **Steps:**
    a. Enter edge case search queries (e.g., extremely long strings, special characters, very common or rare keywords).
    b. Initiate the search.
- **Expected Result:** The system processes edge case queries gracefully without errors.
- **Pass Criteria:**
    a. The system does not crash or become unresponsive.
    b. Appropriate error messages or search results are provided.
    c. The search functionality remains robust and stable.

Test Case 9: Internationalization and Localization

- **Objective:** Ensure the search and match functionalities work correctly with resumes and job descriptions in different languages.
- **Preconditions:** Access to the search and match feature, and a database with resumes in multiple languages.
- **Steps:**
    a. Perform searches and matching operations using keywords and job descriptions in various supported languages.

    b. Verify the accuracy of search results for each language.
- **Expected Result:** The system accurately processes and returns relevant results for different languages.
- **Test Data:** Search queries and job descriptions in various supported languages.
- **Pass Criteria:**
    a. Search and match results are accurate across different languages.
    b. The system supports all listed languages effectively.

Test Case 10: Security and Data Privacy

- **Objective:** Validate the security and data privacy aspects of the search and match functionalities.
- **Preconditions:** Access to the search and match feature with various user roles.
- **Steps:**
    a. Perform searches and access search results with different user roles (e.g., recruiter, admin, guest).
    b. Attempt to access restricted data or perform unauthorized actions.
- **Expected Result:** The system enforces proper security and privacy controls.
- **Test Data:** User accounts with different roles and permissions.
- **Pass Criteria:**
    a. Users can only access data and perform actions according to their roles and permissions.
    b. Unauthorized access and actions are prevented and logged.

Test Case 11: Scalability Testing

- **Objective:** Assess the scalability of the search and match functionalities as the number of candidates and job descriptions increases.
- **Preconditions:** Access to a performance testing environment with a large, scalable database.
- **Steps:**

a. Incrementally increase the number of candidate profiles and job descriptions in the database.
b. Perform search and match operations and monitor system performance.

- **Expected Result:** The system maintains acceptable performance levels as the database scales.
- **Test Data:** Large volumes of candidate profiles and job descriptions.
- **Pass Criteria:**
    a. The system handles increased data volumes without significant performance degradation.
    b. Search and match response times remain within acceptable limits.

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.

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.

a. **Development (20hr – 25hr)**
    a. Integration with RChilli API for Parsing & Indexing: 2hr
    b. Simple Search Implementation: 3hr
    c. Boolean Search Implementation: 4hr
    d. Matching Functionality Implementation: 5hr
    e. Enhancing Search Algorithms for Accuracy: 3-4hr
    f. Unit Test Cases: 2-3hr
    g. Bug Fixing and Optimization: 1-2hr
b. **QA (8hr - 10hr)**
    a. UAT: 2-3hr
    b. Executing Test Cases: 6-7hr

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 Index Method:

https://docs.rchilli.com/kc/internal/c_RChilli_search_match_API_Endpoints_Parse_index

Search Engine Features:

https://docs.rchilli.com/kc/internal/c_RChilli_search_match_Features

Search Engine Response Schema:

https://docs.rchilli.com/kc/internal/c_RChilli_search_match_Schema

Search Engine Error Code:

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

Language Supported:

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

Postman Integration and Sample Code:

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