Apply and Recommend

with RChilli

June 2024 | Version 1.0

Purpose Statement

This document details integrating the **Apply and Recommend** use case leveraging RChilli Search and Match functionality, enabling candidates to upload their resume once to receive job recommendations and apply for jobs with their information filled out automatically.

Disclaimer:

Contents

Introduction

The **Apply and Recommend** feature streamlines the job application process by allowing candidates to upload their resume once to receive job recommendations and then apply for jobs with their information filled out automatically. Utilizing the RChilli Resume Parser and Search and Match APIs, the system automates the extraction of relevant information and matches it with job openings, providing a seamless experience for candidates.

Apply and recommend simplifies job applications by:

- **Uploading Candidate Profiles:** Candidates upload their profiles to the system.
- **Parsing And Indexing Candidate Profiles:** Profiles are processed using the RChilli Search and Match Engine API.
- **Matching Profiles to Jobs:** The system match candidate profiles to job openings using the RChilli Search and Match API.
- **Providing Recommendations:** Candidates receive a list of recommended jobs matching their profiles.
- **Quick Apply:** Candidates can apply for recommended jobs with their information automatically filled out.

Key Components and Flow

1. **Processing Job Applications:**
    a. **Action:** All job applications will be indexed using RChilli Search and Match api.
    b. **Functionality:**
        i. Api will return Structured parsed Json output
        ii. Job data indexed in RChilli
        iii. Store parsed data with index Id for application functionality.
    c. **Response:** Generates a structured JSON output along with an index ID for each job.
2. **Profile Upload:**
    a. **Action:** Candidates upload their resumes (PDF, DOCX, etc.) to the system.
    b. **Response:** The profiles are sent to the server for processing.
3. **Profile Parsing:**
    a. **Action:** The server calls the RChilli Search and Match Engine API for parsing and indexing the candidate's resume.
    b. **Functionality:**
        i. Api will return Structured parsed Json output
        ii. Document Index in RChilli
        iii. Store Parsed data with index Id for application functionality.
    c. **Response:** Resume parsed Json with index Id will be returned.
4. **Resume-to-Job Matching**:
    a. **Action:** The system calls the RChilli Search and Match API with the parsed profile data.
    b. **Functionality:**
        i. The API matches candidate resume to job openings based on skills, experience, and education.
    c. The matching criteria can be defined dynamically using RChilli's configuration.

d. **Response:** The API returns a list of recommended jobs for each candidate profile.

5. **Recommendation Display:**
    a. **Action:** The system displays the list of recommended jobs to the candidate with reference to the index ID returned from the Match API and the relevant data stored in the database.
    b. **Response:** Candidates can review, shortlist, and apply to jobs directly from the recommendation list.
6. **Quick Apply:**
    a. **Action:** Candidates select a recommended job and choose to apply using the Quick Apply feature. Candidate data can be pulled from databased and submitted to new job without need of reparsing, indexing or matching.
    b. **Response:** The system auto-fills the application form with the candidate's information extracted from the resume, allowing for a fast and easy application process.
7. **Data Storage:**
    a. **Action:** The matched data and application details are stored in the database.
    b. **Functionality:** This data can be used for reporting and future reference.

Benefits

- **Time Efficiency:** Significantly reduces the time candidates spend on finding and applying to job openings that match their profiles.
- **User-Friendly:** Improves the user experience by automating the job recommendation and application process.

User Stories

1. **Upload Job**
    - **As a Recruited,** I want to upload my job so that I can receive matching candidate can apply.
2. **Parse And Index Job**
    - **As a system,** I want to parse and index the Job
3. **Store Job Data In Database**
    - **As a system,** I want to store job parsed data with index id into database
4. **Upload Resume**
    - **As a candidate,** I want to upload my resume so that I can get job recommendations.
5. **Parse And Index Resume**
    - **As a system,** I want to parse and index the Job
6. **Store Job Data in Database**
    - **As a system,** I want to store resume parsed data with index id into database
7. **Match Profile to Jobs**
    - **As a system,** I want to match candidate Resume to jobs indexed to provide accurate recommendations.
8. **Display Recommendations**
    - **As a candidate,** I want to see a list of recommended jobs that match my profile.
9. **Quick Apply**
    - **As a candidate,** I want to apply for recommended jobs with my information auto-filled in the application form.

Example Scenario

1. **Scenario**: A candidate named Alex finds a job posting on your company's website and wants to apply.
2. **Action**: Alex clicks on the "Quick Apply" button and uploads their resume.
3. **Response**: The system immediately processes the resume using the Rchilli API, auto-fills the application form with Alex's details, and displays it for review.
4. **Outcome**: Alex reviews the pre-filled information, makes minor adjustments, and submits the application - all within a few seconds.

User Acceptance Testing (UAT) Scenarios

UAT Scenario 1: Successful Job Upload and Parsing

- **Objective:** Verify that a recruiter can upload a job posting and the system successfully parses and indexes it.
- **Preconditions:** Recruiter has a job posting ready for upload.
- **Steps:**
    1. Navigate to the job upload page.
    2. Upload a job posting.
- **Expected Result:** The job posting is uploaded, and the system parses and indexes the job.
- **Acceptance Criteria:**
    - Job upload completes without errors.
    - Parsed job data is received in a structured format.

UAT Scenario 2: Store Job Data in Database

- **Objective:** Confirm that the parsed job data is stored in the database with an index ID.
- **Preconditions:** Successful parsing and indexing of the job posting.
- **Steps:**
    1. Upload a job posting.
    2. Verify that the parsed data is stored in the database with an index ID.
- **Expected Result:** Parsed job data is stored accurately.
- **Acceptance Criteria:**
    - Database entries match the parsed job data.

UAT Scenario 3: Successful Resume Upload and Parsing

- **Objective:** Verify that a candidate can upload their resume and the system successfully parses and indexes it.

- **Preconditions:** Candidate has a resume ready for upload.
- **Steps:**
    1. Navigate to the resume upload page.
    2. Upload a resume.
- **Expected Result:** The resume is uploaded, and the system parses and indexes the resume.
- **Acceptance Criteria:**
    - Resume upload completes without errors.
    - Parsed resume data is received in a structured format.

UAT Scenario 4: Store Resume Data in Database

- **Objective:** Confirm that the parsed resume data is stored in the database with an index ID.
- **Preconditions:** Successful parsing and indexing of the resume.
- **Steps:**
    1. Upload a resume.
    2. Verify that the parsed data is stored in the database with an index ID.
- **Expected Result:** Parsed resume data is stored accurately.
- **Acceptance Criteria:**
    - Database entries match the parsed resume data.

UAT Scenario 5: Profile-to-Job Matching

- **Objective:** Validate that candidate profiles are correctly matched to job openings.
- **Preconditions:** Successful parsing and indexing of both job postings and resumes.
- **Steps:**
    1. Upload job postings and candidate resumes.
    2. Match profiles to jobs using the RChilli API.
- **Expected Result:** Accurate job recommendations are provided.
- **Acceptance Criteria:**

- Recommendations match the candidate's profile and job requirements.

UAT Scenario 6: Display Recommendations

- **Objective:** Ensure that job recommendations are displayed correctly to the candidate.
- **Preconditions:** Successful profile-to-job matching.
- **Steps:**
    1. Upload a resume and match it to job postings.
    2. Display the job recommendations to the candidate.
- **Expected Result:** Job recommendations are displayed accurately.
- **Acceptance Criteria:**
    - Recommendations are relevant and accurate.

UAT Scenario 7: Quick Apply Functionality

- **Objective:** Confirm that candidates can apply to recommended jobs with their information auto filled.
- **Preconditions:** Successful profile-to-job matching and recommendation display.
- **Steps:**
    1. Select a recommended job.
    2. Click the "Quick Apply" button.
    3. Review and submit the auto-filled application form.
- **Expected Result:** Application form is correctly auto-filled with candidate's information.
- **Acceptance Criteria:**
    - Application form fields are accurately filled.
    - Candidate can make edits if necessary.

Test Cases

Test Case 1: Job Upload Test

- **Objective:** Verify that a job posting can be uploaded successfully.
- **Preconditions:** Recruiter has a job posting ready.
- **Steps:**
    1. Navigate to the job upload page.
    2. Upload a job posting.
- **Expected Result:** Job posting upload complete without errors.
- **Pass Criteria:** Job is uploaded and available for processing.

Test Case 2: Parse and Index Job Test

- **Objective:** Ensure the job posting is parsed and indexed successfully by the RChilli API.
- **Preconditions:** Job posting is uploaded.
- **Steps:**
    1. Upload a job posting.
    2. Verify the API call to RChilli.
    3. Confirm receipt of a structured JSON response.
- **Expected Result:** JSON response is received with parsed job data.
- **Pass Criteria:** API call completes successfully and returns valid JSON.

Test Case 3: Store Job Data in Database Test

- **Objective:** Verify that parsed job data is correctly stored in the database.
- **Preconditions:** Successful job parsing and indexing.
- **Steps:**
    1. Upload a job posting.
    2. Verify that the parsed data is stored in the database with an index ID.
- **Expected Result:** Parsed job data is stored accurately in the database.

- **Pass Criteria:** Database entries match the parsed job data.

Test Case 4: Resume Upload Test

- **Objective:** Verify that a resume can be uploaded successfully.
- **Preconditions:** Candidate has a resume ready.
- **Steps:**
    1. Navigate to the resume upload page.
    2. Upload a resume.
- **Expected Result:** Resume upload completes without errors.
- **Pass Criteria:** Resume is uploaded and available for processing.

Test Case 5: Parse and Index Resume Test

- **Objective:** Ensure the resume is parsed and indexed successfully by the RChilli API.
- **Preconditions:** Resume is uploaded.
- **Steps:**
    1. Upload a resume.
    2. Verify the API call to RChilli.
    3. Confirm receipt of a structured JSON response.
- **Expected Result:** JSON response is received with parsed resume data.
- **Pass Criteria:** API call completes successfully and returns valid JSON.

Test Case 6: Store Resume Data in Database Test

- **Objective:** Verify that parsed resume data is correctly stored in the database.
- **Preconditions:** Successful resume parsing and indexing.
- **Steps:**
    1. Upload a resume.
    2. Verify that the parsed data is stored in the database with an index ID.
- **Expected Result:** Parsed resume data is stored accurately in the database.

- **Pass Criteria:** Database entries match the parsed resume data.

Test Case 7: Profile-to-Job Matching Test

- **Objective:** Validate that profiles are matched to job openings accurately.
- **Preconditions:** Successful parsing and indexing of job postings and resumes.
- **Steps:**
    1. Upload job postings and resumes.
    2. Match profiles to jobs using the RChilli API.
- **Expected Result:** Accurate job recommendations are provided.
- **Pass Criteria:** Recommendations match candidate profiles and job requirements.

Test Case 8: Recommendation Display Test

- **Objective:** Confirm that job recommendations are displayed correctly to the candidate.
- **Preconditions:** Successful profile-to-job matching.
- **Steps:**
    1. Upload a resume and match it to job postings.
    2. Display the job recommendations to the candidate.
- **Expected Result:** Job recommendations are displayed accurately.
- **Pass Criteria:** Recommendations are relevant and accurate.

Test Case 9: Quick Apply Test

- **Objective:** Ensure the Quick Apply functionality works correctly.
- **Preconditions:** Successful profile parsing, job matching, and recommendation display.
- **Steps:**
    1. Select a recommended job.
    2. Click the "Quick Apply" button.
    3. Verify the auto-filled application form.
- **Expected Result:** Application form is auto-filled accurately with candidate's information.

- **Pass Criteria:** Auto-filled fields are correct and editable if needed.

Test Case 10: Data Storage Test

- **Objective:** Verify that matched data and application details are correctly stored in the database.
- **Preconditions:** Successful profile parsing, job matching, and application submission.
- **Steps:**
    1. Upload a resume and match it to job postings.
    2. Submit the application using Quick Apply.
    3. Check the database for stored data.
- **Expected Result:** Matched data and application details are stored accurately.
- **Pass Criteria:** Database entries match the parsed and matched data.

Test Case 11: Error Handling for Job Upload

- **Objective:** Verify that the system handles errors during job upload appropriately.
- **Preconditions:** Recruiter attempts to upload a job posting with incorrect format or missing fields.
- **Steps:**
    1. Navigate to the job upload page.
    2. Upload a job posting with incorrect format or missing required fields.
- **Expected Result:** The system displays an appropriate error message.
- **Pass Criteria:** Error message accurately describes the issue and provides guidance on resolving it.

Test Case 12: Error Handling for Resume Upload

- **Objective:** Verify that the system handles errors during resume upload appropriately.
- **Preconditions:** Candidate attempts to upload a resume with an unsupported file format or exceeding the file size limit.
- **Steps:**

1. Navigate to the resume upload page.
2. Upload a resume with an unsupported file format or exceeding the file size limit.

- **Expected Result:** The system displays an appropriate error message.
- **Pass Criteria:** Error message accurately describes the issue and provides guidance on resolving it.

Test Case 13: Resume Upload with Multiple File Types

- **Objective:** Verify that the system can handle different resume file types (PDF, DOCX, etc.).
- **Preconditions:** Candidate has resumes in various supported formats.
- **Steps:**
    1. Navigate to the resume upload page.
    2. Upload resumes in different supported formats (PDF, DOCX, etc.).
- **Expected Result:** The system successfully uploads and parses resumes in all supported formats.
- **Pass Criteria:** Resumes are parsed and indexed correctly regardless of file format.

Test Case 14: Job Search and Filter

- **Objective:** Verify that candidates can search, and filter job recommendations based on different criteria.
- **Preconditions:** Candidate has uploaded a resume and received job recommendations.
- **Steps:**
    1. Navigate to the job recommendations page.
    2. Use search and filter options to refine job recommendations (e.g., by location, job type, salary).
- **Expected Result:** The system displays job recommendations that match the search and filter criteria.
- **Pass Criteria:** Recommendations are relevant and accurately reflect the search and filter criteria.

Test Case 15: Data Privacy and Security

- **Objective:** Verify that the system ensures data privacy and security for candidate information.
- **Preconditions:** Candidate data is stored in the database.
- **Steps:**
    1. Attempt unauthorized access to candidate data.
    2. Verify the security measures in place.
- **Expected Result:** Unauthorized access is prevented and data privacy is maintained.
- **Pass Criteria:** Data access is secure, and privacy measures are effective.

Test Case 16: Performance Testing for Resume Parsing

- **Objective:** Assess the system's performance during the parsing of large volumes of resumes.
- **Preconditions:** System has a queue of multiple resume uploads.
- **Steps:**
    1. Simultaneously upload multiple resumes.
    2. Measure the time taken to parse and index each resume.
- **Expected Result:** System processes resume within an acceptable time frame without errors.
- **Pass Criteria:** Parsing and indexing times are within acceptable limits, and system performance is stable.

Test Case 17: User Interface Responsiveness

- **Objective:** Verify the responsiveness of the user interface across different devices and screen sizes.
- **Preconditions:** Access to multiple devices with different screen sizes.
- **Steps:**
    1. Navigate to the job upload and resume upload pages on various devices (desktop, tablet, mobile).

2. Perform actions such as uploading resumes and applying for jobs.

- **Expected Result:** The user interface adapts and responds appropriately on all devices.
- **Pass Criteria:** UI is user-friendly and fully functional across all tested devices.

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 that 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 a 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 (14hr – 17hr)**
    a. Upload Profile - 1hr
    b. Parse and Index using RChilli API – 1hr
    c. Read JSON and display recommendations on UI – 6hr
    d. Store information in Database – 2-3hr
    e. Implement Quick Apply functionality – 2-3hr
    f. Unit Test cases – 2-3hr
    g. Bug fixing and optimization – 2-3hr
2. **QA – (5hr -7hr)**
    a. UAT – 1-1.5hr
    b. Executing Test cases – 4-5.5hr

Appendix

Search and Match Api Details:

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

Search and Match Parse and Index API Details:

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

Search and Match, Match API Details:

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

Get API key and Endpoints:

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

Postman Integration and Sample Code:

https://documenter.getpostman.com/view/6003669/Szmh1GQX?version=latest#0b611a07-dd2d-4737-8b0c-68b14d17746f

Search and Match Response Example:

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

Search Engine Dynamic Weightage settings:

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

Search and Match API Error Code:

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

Language Supported:

https://www.rchilli.com/languages