Quick Apply

with RChilli

June 2024 | Version 1.0

Purpose Statement

This document details integrating Quick Apply use case leveraging Resume parsing.

Disclaimer:

Contents

Key Components and Flow 5

Benefits 6

Example Scenario 7

User Acceptance Testing (UAT) 8

UAT Scenarios 8

Test Cases 10

Test Case 1: Resume Upload Test 10

Test Case 2: API Call Test 10

Test Case 3: JSON Data Processing Test 10

Test Case 4: Unsupported File Format Handling 11

Test Case 5: Data Storage Test 11

Test Case 6: Candidate Editable Autofill Fields 11

Test Case 7: Resume Size Limit Handling 12

Test Case 8: Resume with Non-Standard Formatting 12

Test Case 9: Duplicate Field Handling 12

Test Case 10: Missing Data Handling 13

Test Case 11: Special Characters Handling 13

Test Case 12: Multiple Language Support 14

Test Case 13: Network Failure Handling 14

Test Case 14: Retry Mechanism 14

Test Case 15: API Rate Limiting 15

Test Case 16: Cross-Browser Compatibility 15

Miscellaneous 16

Integration Estimation 16

Appendix 17

Introduction

The **Quick Apply** feature streamlines the job application process by allowing candidates to apply with their resume in under 5 seconds. Utilizing the Rchilli Resume Parser API, the system automates the extraction of relevant information from the uploaded resume, significantly reducing the time and effort required for candidates to complete application forms.

Quick Apply simplifies job applications by:

1. **Uploading Resume**: Candidates upload their resume directly to the application portal.
2. **Parsing Resume**: The system instantly processes the uploaded resume using the Rchilli Resume Parser API.
3. **Auto filling Information**: Extracted data is used to auto-populate application form fields or stored in the database, expediting the application process.

Key Components and Flow

1. **Candidate Interaction**: Candidates interact with the job application interface and use the Quick Apply button to upload their resume. This initiates the application process with minimal input required beyond selecting the resume file.
2. **Resume Upload**:
    - **Action**: The candidate selects their resume file (PDF, DOCX, etc.) and uploads it.
    - **Response**: The file is sent to the server for processing.
3. **Resume Parsing**:
    - **Action**: The server calls the Rchilli Resume Parser API with the uploaded resume.
    - **Functionality**: The API processes the resume, extracting key information such as:
        - Personal details (name, contact information)
        - Professional summary
        - Employment history
        - Education
        - Skills
    - **Response**: The API returns this data in a structured JSON format.
4. **Autofill and Display**:
    - **Action**: The system reads the JSON data and populates the application form fields with the parsed information.
    - **Optional**: The candidate can review and edit the auto-filled information, if necessary, before submitting based on UI.
    - **Response**: Provides a preview or confirmation of the auto-filled form to the candidate.
5. **Data Storage**:
    - **Action**: The parsed resume data is stored in the database, associated with the job application.
    - **Functionality**: This data can be used for backend processing, reporting, and future reference.

Benefits

1. **Time Efficiency**: Drastically reduces the time candidates spend on filling out job applications by automating data entry.
2. **User-Friendly**: Enhances the user experience by reducing the complexity of job applications to a simple resume upload.

User Stories

1. **Upload Resume**
    - **As a candidate**, I want to upload my resume so that I can apply for a job without manually entering my information.
2. **Parse Resume**
    - **As a system**, I need to parse the uploaded resume to extract relevant data and present it in a structured format.
3. **Autofill Application Form**
    - **As a candidate**, I want the application form to be automatically filled with my resume details so that I can quickly review and submit my application.
4. **Store Parsed Data**
    - **As a system**, I need to store the parsed resume data in the database to maintain accurate records and facilitate the application process.

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)

The **Quick Apply** UAT plan ensures the feature meets user requirements and business goals. It validates that candidates can apply for jobs efficiently by uploading their resumes, which are then processed to populate application forms or stored in the database.

UAT Scenarios

UAT Scenario 1: Successful Resume Upload and Parsing

- **Objective**: Verify that a candidate can upload a resume and the system successfully parses it.
- **Preconditions**: Candidate has a resume file ready for upload.
- **Steps**:
    1. Navigate to the job application page.
    2. Click the "Quick Apply" button.
    3. Upload a resume file.
- **Expected Result**: The resume is uploaded, and the system receives a JSON response with the parsed data.
- **Acceptance Criteria**:
    - Resume upload completes without errors.
    - JSON response includes correctly parsed fields.

UAT Scenario 2: Autofill Application Form

- **Objective**: Validate that the application form is correctly auto filled using the parsed resume data.
- **Preconditions**: Successful parsing of resume.
- **Steps**:
    1. Upload a resume using the "Quick Apply" feature.
    2. Observe the auto-filled fields in the application form.

3. Review and submit the application.

- **Expected Result**: The application form fields are populated with the correct data from the resume.
- **Acceptance Criteria**:
    - All relevant fields are accurately filled.
    - Candidate can make edits if necessary.

UAT Scenario 3: Error Handling for Unsupported File Formats

- **Objective**: Ensure the system handles unsupported file formats gracefully.
- **Preconditions**: Candidate has a file in an unsupported format.
- **Steps**:
    1. Click the "Quick Apply" button.
    2. Attempt to upload a file in an unsupported format (e.g., TXT).
- **Expected Result**: The system displays an error message indicating the file format is unsupported.
- **Acceptance Criteria**:
    - Candidate is informed of the unsupported format.
    - Candidate is prompted to upload a valid resume file.

UAT Scenario 4: Data Storage Verification

- **Objective**: Confirm that the parsed data is correctly stored in the database.
- **Preconditions**: Successful parsing and form submission.
- **Steps**:
    1. Upload a resume and submit the application.
    2. Verify the stored data in the database.
- **Expected Result**: All parsed data is stored accurately and linked to the job application.
- **Acceptance Criteria**:
    - Database entries match the parsed data.
    - Data integrity is maintained.

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 job application form.
    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: API Call Test

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

Test Case 3: JSON Data Processing Test

- **Objective**: Validate that the JSON data is correctly processed and can populate the application form.
- **Preconditions**: Successful API response with JSON data.
- **Steps**:

1. Upload a resume and process the JSON response.
2. Check that application form fields are populated.

- **Expected Result**: Fields are accurately filled based on the JSON data.
- **Pass Criteria**: Application form fields are correctly auto-filled.

Test Case 4: Unsupported File Format Handling

- **Objective**: Ensure system handles unsupported file formats appropriately.
- **Preconditions**: Candidate attempts to upload an unsupported file format.
- **Steps**:
    1. Click the upload button.
    2. Try to upload an unsupported file format.
- **Expected Result**: System displays an error message.
- **Pass Criteria**: Error message is shown, and upload is rejected.

Test Case 5: Data Storage Test

- **Objective**: Verify that parsed resume data is correctly stored in the database.
- **Preconditions**: Successful resume parsing and form submission.
- **Steps**:
    1. Upload and parse a resume.
    2. Submit the application.
    3. Check the database for stored data.
- **Expected Result**: Parsed data is stored accurately in the database.
- **Pass Criteria**: Database entries match the parsed resume data.

Test Case 6: Candidate Editable Autofill Fields

- **Objective**: Ensure candidates can edit auto-filled fields before submitting.
- **Preconditions**: Application form is auto filled.
- **Steps**:

1. Upload a resume and observe the auto-filled form.
2. Edit one or more auto-filled fields.
3. Submit the application.

- **Expected Result**: Edited fields are saved correctly.
- **Pass Criteria**: Edited data is reflected in the final application and stored data.

Test Case 7: Resume Size Limit Handling

- **Objective**: Ensure system handles large resume files properly.
- **Preconditions**: Candidate attempts to upload a large resume file.
- **Steps**:
    1. Try to upload a resume file exceeding the maximum size limit.
- **Expected Result**: System displays an error message regarding file size.
- **Pass Criteria**: Error message is shown, and upload is prevented.

Test Case 8: Resume with Non-Standard Formatting

- **Objective:** Verify that the system can parse resumes with non-standard formatting.
- **Preconditions:** Candidate has a resume with non-standard formatting (e.g., unconventional fonts, layouts).
- **Steps:**
    1. Navigate to the job application form.
    2. Click the upload button.
    3. Select and upload the non-standard formatted resume file.
- **Expected Result:** The resume is parsed correctly despite the non-standard formatting.
- **Pass Criteria:** Key information (e.g., personal details, employment history, skills) is extracted accurately.

Test Case 9: Duplicate Field Handling

- **Objective:** Ensure the system handles duplicate fields in resumes correctly.

- **Preconditions:** Candidate has a resume with duplicate fields (e.g., two sections labeled "Skills").
- **Steps:**
    1. Upload a resume with duplicate fields.
    2. Verify the auto-filled application form.
- **Expected Result:** Duplicate fields are handled gracefully, and data is populated without errors.
- **Pass Criteria:** No duplication errors, and the form displays the correct data.

Test Case 10: Missing Data Handling

- **Objective:** Validate how the system handles resume with missing data.
- **Preconditions**: Candidate has a resume with missing sections (e.g., no education history).
- **Steps:**
    1. Upload a resume with missing data.
    2. Verify the auto-filled application form.
- **Expected Result:** The form is populated with available data, and fields with missing data are left empty or marked as required.
- **Pass Criteria:** The application form correctly reflects the available data without errors.

Test Case 11: Special Characters Handling

- **Objective:** Ensure the system can process resumes containing special characters.
- **Preconditions:** Candidate has a resume with special characters (e.g., accented characters, symbols).
- **Steps:**
    1. Upload a resume containing special characters.
    2. Verify the auto-filled application form.
- **Expected Result:** Special characters are displayed correctly in the form.
- **Pass Criteria:** All special characters are accurately represented in the application form.

Test Case 12: Multiple Language Support

- **Objective:** Verify that the system can parse resumes written in different languages.
- **Preconditions:** Candidate has resumes in different languages supported by RChilli.
- **Steps:**
    1. Upload resumes in various supported languages.
    2. Verify the auto-filled application forms for each language.
- **Expected Result:** The forms are populated correctly with data from resumes in different languages.
- **Pass Criteria:** Accurate data extraction and population for each language**.**

Test Case 13: Network Failure Handling

- **Objective:** Ensure the system handles network failures gracefully during the upload process.
- **Preconditions:** Simulate a network failure during resume upload.
- **Steps:**
    1. Start uploading a resume.
    2. Interrupt the network connection.
- **Expected Result:** The system displays an appropriate error message and allows the candidate to retry the upload.
- **Pass Criteria:** The system does not crash and provides clear instructions for retrying.

Test Case 14: Retry Mechanism

- **Objective:** Validate the retry mechanism for resume upload failures.
- **Preconditions:** Resume upload has previously failed.
- **Steps:**
    1. Attempt to re-upload the same resume.
- **Expected Result:** The system successfully processes the resume on the retry.
- **Pass Criteria:** Resume upload and parsing succeed on the retry attempt.

Test Case 15: API Rate Limiting

- **Objective:** Ensure the system handles API rate limiting gracefully.
- **Preconditions**: Exceed the RChilli API call limit.
- **Steps:**
    1. Rapidly upload multiple resumes to trigger rate limiting.
- **Expected Result:** The system handles rate limiting without crashing and provides appropriate feedback to the user.
- **Pass Criteria:** The system displays a rate limit error message and prevents further uploads until the rate limit resets.

Test Case 16: Cross-Browser Compatibility

- **Objective:** Verify that the resume upload and parsing functionality works across different web browsers.
- **Preconditions:** Access to various web browsers (e.g., Chrome, Firefox, Safari, Edge).
- **Steps:**
    1. Open the job application form in different web browsers.
    2. Upload a resume in each browser.
- **Expected Result:** The resume upload and parsing work correctly in all browsers.
- **Pass Criteria:** Consistent functionality and correct data parsing across all tested browsers.

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.

1. **Development** (14hr – 17hr)
    - Upload Resume - 1hr
    - Parse using RChilli AI – 1hr
    - Read Json and populate on UI – 6hr
    - Store information in Database – 2-3hr
    - Unit Test cases – 2-3hr
    - Bug fixing and optimization – 2-3hr

2. **QA** – (5hr -7hr)
    - UAT – 1-1.5hr
    - Executing Test cases – 4-5.5hr

Appendix

Resume Parser Api Details:

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

Get API key and Endpoints:

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 Configuration settings:

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

Resume Parser Error Code:

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

Language Supported:

https://www.rchilli.com/languages