Unbiased or Redaction of Candidate Information

with RChilli

June 2024 | Version 1.0

Purpose Statement

This document details integrating **Unbiased / Redaction of Candidate Information** use case leveraging RChilli Redactor API.

Disclaimer:

Contents

Introduction

The **Unbiased/Redaction of Candidate Information** feature improves nonbiased hiring practices by automatically creating redacted resumes.

It is also useful for branding resumes and removing contact personal and educational information for staffing and recruiting firms.

The system gets the resume from the candidate, redacts it using the RChilli Redaction API, stores the redacted document in the database, and provides the redacted resume to recruiters when they review candidates for job openings.

Impact:

- Reduces bias-related incidents by 80%
- Speeds up the candidate review process by up to 70%

Key Components and Flow

1. **Resume Reception:** The candidate interacts with the interface and uploads their resume.
    a. **Action:** The candidate submits their resume.
    b. **Response:** The resume is uploaded to the server for processing.
2. **Redaction Using RChilli API:**
    a. **Action:** The system sends the resume to the RChilli redaction API.
    b. **Response:** The API will process the resume and return a redacted version of the resume. The redaction will be applied to fields, as mentioned below:
        i. Candidate Name
        ii. Candidate Email
        iii. Address
        iv. Phone Number
        v. Employment and Education Details
3. **Database Storage:**
    a. **Action:** The redacted resume is stored in the database.
    b. **Response:** The document is saved and indexed for future reference.
4. **Return Redacted Document:**
    a. **Action**: Recruiters access the redacted resumes when reviewing candidates for job openings.
    b. **Response:** Recruiters receive the redacted resumes for unbiased candidate evaluation.

Benefits

1. **Nonbiased Hiring:** Reduces bias-related incidents by 80%.
2. **Efficiency:** Speeds up the candidate review process by up to 70%.
3. **Branding:** Helps staffing and recruiting firms maintain consistent resume formats and remove unnecessary contact information.

User Stories

1. **Upload Resume**
    a. As a candidate, I want to upload my resume for redaction so that my personal information can be removed for unbiased hiring.
2. **Redact Resume Information**
    a. As a system, I need to redact sensitive information from the uploaded resume to ensure unbiased hiring.
3. **Store Redacted Resume**
    a. As a system, I need to store the redacted resume in the database to keep a record of the redacted information.
4. **Retrieve Redacted Resume**
    a. As a recruiter, I want to receive redacted resumes to review candidates without bias.

Example Scenario

1. **Scenario:** A candidate named Sam submits their resume through the job application portal.
2. **Action:** Sam's resume is automatically sent to the RChilli redaction API.
3. **Response:** The redacted resume is stored in the database and sent back to Sam.
4. **Outcome**: When a recruiter accesses the system to review candidates for a job opening, they receive the redacted version of the resume.

User Acceptance Testing (UAT)

The **Unbiased/Redaction of Candidate Information** 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 redacted properly, stored in the database and return the same redacted resume.

UAT Scenarios

UAT Scenario 1: Resume Submission and Redaction

- **Objective:** Verify that a resume can be submitted and redacted successfully.
- **Preconditions:** Candidate has a resume file ready for submission.
- **Steps:**
    a. Submit the resume through the job application portal.
    b. Monitor the process to ensure the resume is sent to the RChilli redaction API.
    c. Check the response from the API.
    d. Verify the redacted resume is stored in the database.
    e. Verify the redacted resume is returned to the candidate.
- **Expected Result:** Resume is submitted, redacted, stored, and returned without errors.
- **Acceptance Criteria:**
    a. Resume submission completes without errors.
    b. Redacted resume includes correctly redacted fields.
    c. Redacted resume is stored in the database.
    d. Redacted resume is returned to the candidate.

UAT Scenario 2: JSON Data Storage

- **Objective:** Verify that the redacted JSON data can be stored in the database.
- **Preconditions:** Redacted JSON data from the RChilli API.
- **Steps:**

a. Receive the JSON response from the RChilli API.
b. Store the data in the database.

- **Expected Result:** Data is stored accurately in the ATS database.
- **Acceptance Criteria:**
    a. Data is stored without errors.
    b. Database records are accurate.

UAT Scenario 3: Error Handling for Unsupported Formats

- **Objective:** Verify that the system handles unsupported resume formats gracefully.
- **Preconditions:** Resume with an unsupported format.
- **Steps:**
    a. Submit a resume with an unsupported format through the job application portal.
    b. Monitor the process to see how the system handles the format.
- **Expected Result:** Appropriate error message is generated.
- **Acceptance Criteria:**
    a. System does not crash.
    b. Clear error messages are provided.

UAT Scenario 4: Performance Testing for High Volume Resumes

1. **Objective:** Ensure the system can handle a high volume of resume submissions efficiently.
2. **Preconditions**: Multiple resumes submitted to the system.
3. **Steps:**
    a. Submit multiple resumes through the job application portal.
    b. Monitor processing time and system response.
4. **Expected Result:** All resumes are processed within acceptable time limits.
5. **Acceptance Criteria:**
    a. Processing time is within acceptable limits.
    b. System performance remains stable.

Test Cases

Test Case 1: Resume Submission Test

- **Objective:** Verify that a resume can be submitted successfully.
- **Preconditions:** Candidate has a resume file ready for submission.
- **Steps:**
    a. Submit the resume through the job application portal.
    b. Monitor the process to ensure the resume is sent to the RChilli redaction API.
- **Expected Result:** Resume submission complete without errors.
- **Pass Criteria:** Resume is submitted and sent to the RChilli redaction API successfully.

Test Case 2: Redaction API Response Test

- **Objective:** Verify that the RChilli redaction API returns the redacted resume successfully.
- **Preconditions:** Resume submitted and processed by the RChilli redaction API.
- **Steps:**
    a. Submit the resume through the job application portal.
    b. Monitor the API response.
- **Expected Result:** The API returns a redacted version of the resume.
- **Pass Criteria:** Redacted resume includes correctly redacted fields.

Test Case 3: Redacted Data Storage Test

- **Objective:** Verify that the redacted JSON data can be stored in the database.
- **Preconditions**: Redacted JSON data from the RChilli API.
- **Steps:**
    a. Receive the JSON from the RChilli API & Store the data in the database.
- **Expected Result:** Data is stored accurately in the ATS database.
- **Pass Criteria:** Data is stored without errors, and database records are accurate.

Test Case 4: Unsupported Resume Format Handling Test

- **Objective:** Verify that the system handles unsupported resume formats gracefully.
- **Preconditions:** Resume with an unsupported format.
- **Steps:**
    a. Submit a resume with an unsupported format through the job application portal.
    b. Monitor the process to see how the system handles the format.
- **Expected Result:** Appropriate error message is generated.
- **Pass Criteria:** System does not crash, and clear error messages are provided.

Test Case 5: High Volume Resume Processing Test

1. **Objective:** Ensure the system can handle a high volume of resume submissions efficiently.
2. **Preconditions:** Multiple resumes submitted to the system.
3. **Steps:**
    a. Submit multiple resumes through the job application portal.
    b. Monitor processing time and system response.
4. **Expected Result:** All resumes are processed within acceptable time limits**.**
5. **Pass Criteria:** Processing time is within acceptable limits, and system performance remains stable.

Test Case 6: Resume Size Limit Handling

- **Objective**: Ensure system handles large resume files properly.
- **Preconditions**: Candidate attempts to upload a large resume file.
- **Steps**:
    a. 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 7: Redacted Information Accuracy Test

- **Objective**: Verify that all personal information fields are accurately redacted. .
- **Preconditions**: Candidate resumes with various formats and personal information.
- **Steps**:
    a. Submit multiple resumes with personal information fields (Name, Email, Address, Phone Number, Employment, and Education Details).
    b. Monitor the redaction process and verify the redacted fields.
- **Expected Result**: All specified personal information fields are redacted accurately.
- **Pass Criteria**: Each redacted field (Name, Email, Address, Phone Number, Employment, and Education Details) is redacted correctly in the resume.

Test Case 8: Data Retrieval Test

- **Objective**: Verify that the redacted resume can be retrieved from the database correctly.
- **Preconditions**: Redacted resume stored in the database.
- **Steps**:
    a. Store a redacted resume in the database.
    b. Retrieve the redacted resume from the database.
- **Expected Result**: The retrieved resume matches the redacted resume stored.
- **Pass Criteria**: The retrieved redacted resume is identical to the stored redacted resume.

Test Case 9: Format Preservation Test

- **Objective**: Verify that the original formatting of the resume is preserved after redaction.
  Preconditions: Candidate resumes with varied formatting.
- **Steps**:
    a. Submit resumes with different formats (e.g., PDF, DOCX) for redaction.
    b. Compare the format of the original and redacted resumes.

- **Expected Result**: The formatting of the resumes should remain consistent after redaction.
- **Pass Criteria**: Original formatting is preserved in the redacted resume.

Test Case 10: API Timeout Handling

- **Objective**: Verify that the system handles API timeouts gracefully.
- **Preconditions**: Simulate a slow response or timeout from the RChilli redaction API.
- **Steps**:
    a. Submit a resume during an API timeout simulation.
    b. Monitor the system's handling of the timeout.
- **Expected Result**: The system should handle the timeout gracefully and provide an appropriate error message.
- **Pass Criteria**: The system does not crash, and a clear timeout error message is shown to the user.

Test Case 11: Multi-Language Redaction Test

- **Objective**: Verify that the redaction process works correctly for resumes in different languages. **Preconditions**: Resumes in multiple languages supported by the API.
- **Steps**:
    a. Submit resumes in various languages.
    b. Monitor the redaction process for each language.
- **Expected Result**: Personal information fields are redacted correctly for all supported languages.
- **Pass Criteria**: All personal information fields are accurately redacted, regardless of the resume's language.

Test Case 12: API Error Response Handling

- **Objective**: Verify that the system handles different types of errors returned by the RChilli redaction API.
- **Preconditions**: Simulate various error responses from the RChilli redaction API.
- **Steps**:
    a. Submit resumes to trigger different API error responses (e.g., invalid API key, Expired date etc.)
    b. Monitor the system's handling of these errors.
- **Expected Result**: The system should handle errors gracefully and provide appropriate error messages.
- **Pass Criteria**: The system does not crash, and clear error messages corresponding to each error type are displayed.

Test Case 13: Database Integrity Test

- **Objective**: Verify that storing redacted resumes does not compromise database integrity.
- **Preconditions**: Database with multiple redacted resumes.
- **Steps**:
    a. Store multiple redacted resumes in the database.
    b. Verify the integrity of the database records.
- **Expected Result**: All records should be stored accurately without corruption.
- **Pass Criteria**: Database records are accurate, and no data corruption is observed.

Note: - Please create the other error code test cases according to the Redactor API, click here

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 (20hr – 25hr)**
    a. Resume Upload: 2hr
    b. Resume Redaction using RChilli Redaction API: 4hr
    c. Storing Redacted Resume: 3-4hr
    d. Returning Redacted Resume: 2-3hr
    e. Unit Test Cases: 2-3hr
    f. Bug Fixing and Optimization: 3-4hr
2. **QA – (7hr -10hr)**
    a. UAT: 2-3hr
    b. Executing Test Cases: 5-7hr

Appendix

Redactor API Details:

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

Get API key and Endpoints:

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

Mask Field List:

https://docs.rchilli.com/kc/Redact_API#concept_i5t_2y2_xsb__mask

Sample Redacted Resumes:

https://docs.rchilli.com/kc/Redact_API#concept_i5t_2y2_xsb__section_jdn_hl2_zsb

Postman Integration and Sample Code:

https://documenter.getpostman.com/view/6003669/Szmh1GQX?version=latest#447d7ac9-cd65-4053-b793-4927adfb3b07

Redactor API Error Code:

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

Language Supported:

https://www.rchilli.com/languages