# Candidate Insights

## Explainable AI-Based Resume Parsing and Job Matching

---

**Detailed Technical Documentation**

Presented by
**Sara Khirfan**

Institution
**HTU, ICT Upskilling Program, YGA, Gen AI Track**

# Table of Contents

# 1.0 Introduction

## 1.1 Problem Statement and Project Scenario

The modern hiring process suffers from inefficiency; managers spend excessive time manually reviewing resumes, often resulting in inconsistent evaluations, slow decisions, and unintended bias. *Candidate Insights* solves this by using an intelligent ATS to automate the initial screening while ensuring fairness and transparency. ***It supports -not replaces- human decision-making by offering structured insights, quantified skill evaluations, and clear scoring explanations, enabling faster and more consistent hiring without compromising quality.***

## 1.2 Target Users and Use Cases

The ***primary users are HR teams and hiring managers in small to medium organizations***. They often handle 30–100 applicants per role and need quick, reliable filtering without specialized recruiters. Recruitment agencies also benefit by ensuring consistent evaluations across multiple client roles. ***Secondary users include department heads involved in technical hiring who rely on brief, high-level summaries to understand key strengths and gaps without reviewing full resumes.***

## 1.3 Core Assumptions and Constraints

The design assumes users can provide clear job descriptions with explicit requirements for skills, experience levels, and education qualifications. ***The system is constrained to PDF resume formats, which covers approximately 90% of professional submissions.*** The architecture relies on ***OpenAI's API availability and pricing structure***. The system targets batch processing rather than real-time screening during active interviews. Privacy constraints require all personally identifiable information to remain client-side with visual masking rather than server-side encryption.

## 1.4 System Methodology & User Flow

The interaction follows sequential, structured flow. Users start by uploading one or multiple PDF resumes through drag-and-drop. The system then validates each file, accepting real resumes and rejecting unrelated documents. Next, in Job Management, the user adds a job description by pasting text or uploading a PDF and naming the role. After that, they choose which job to match with the uploaded resumes. Once the matching process starts, progress is

shown as the AI evaluates candidates. Results appear in a sortable dashboard highlighting the top matches with detailed skill, experience, and education insights. Finally, users can export a full PDF report or open individual candidate cards to view the resume, reasoning, and chatbot interaction.

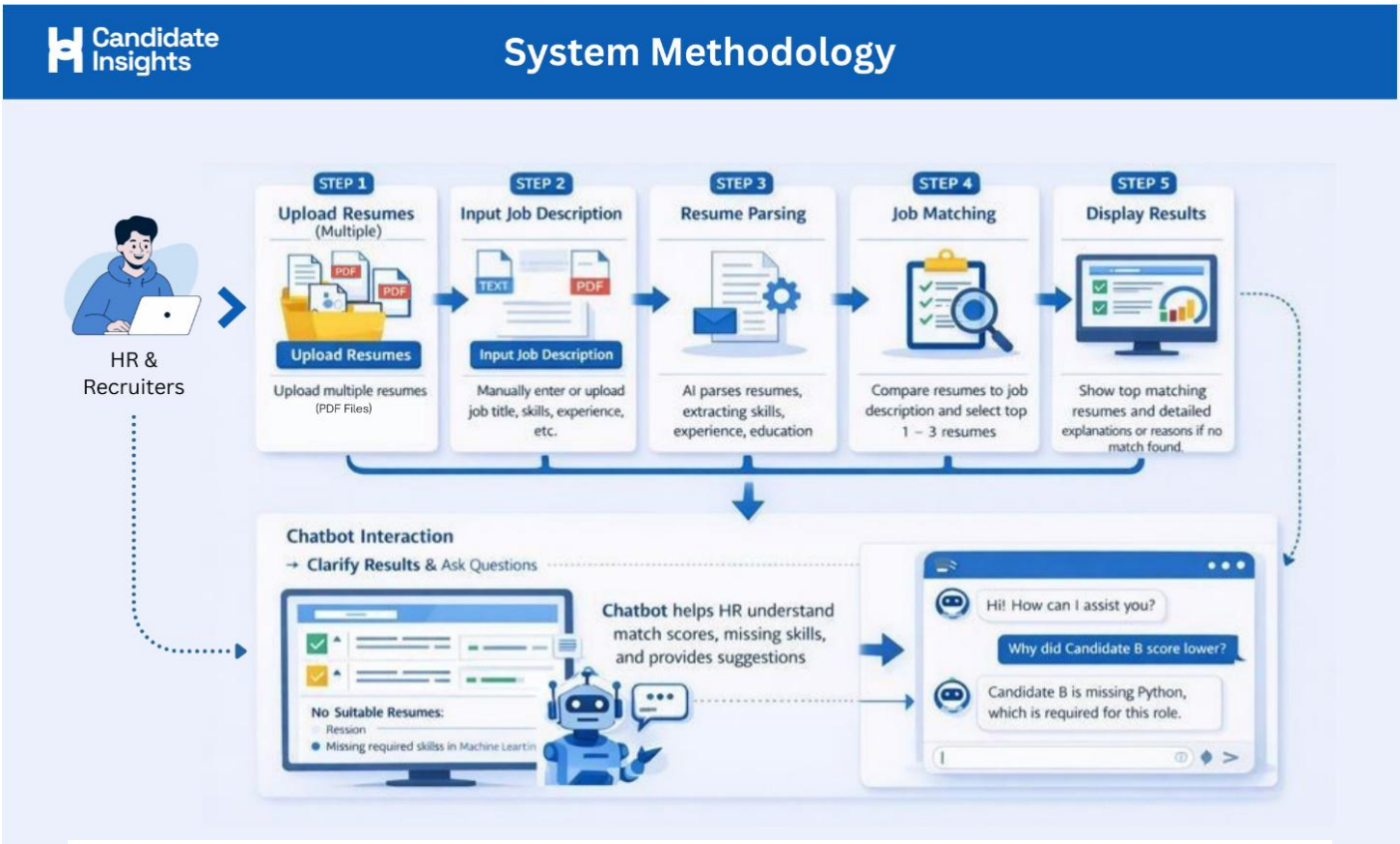


*Figure 1: System Methodology*

## 2.0 System Design

### 2.1 Architecture Diagram

Figure 2 presents the system architecture of *Candidate Insights* system, designed using a clear, sequential pipeline. The process starts with user interactions through a web interface and Flask API, followed by document validation, text extraction, and normalization to ensure input quality. Structured prompts are then applied to extract resume and job requirement data and to calculate matching scores using the selected GPT-4o-mini model via the OpenAI API. The extracted and scored results are stored in a structured data layer, while additional safety, validation, and privacy controls verify score calculations, confidence levels, and sensitive information handling. The architecture is supported by an evaluation module and

comprehensive logging to ensure reliability, traceability, and transparency. A connected chatbot component provides explainable access to matching results for end users.

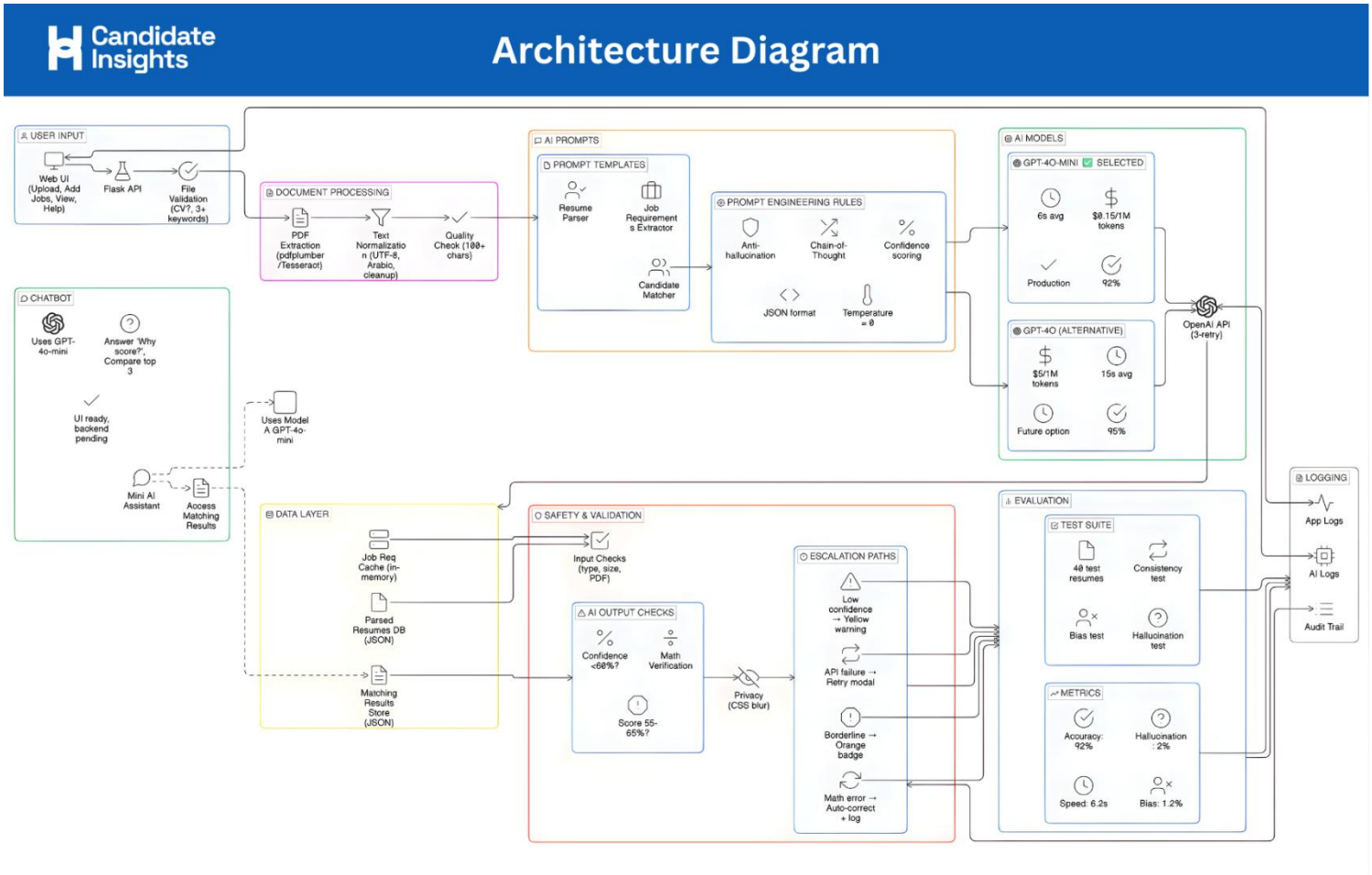


*Figure 2: Architecture Diagram*

### 2.2 Data Flow and Processing Pipeline

***Resume Upload:*** Uploaded PDFs are validated by checking for resume keywords like "experience" and "skills." Text is extracted directly or via OCR if needed, then sent to GPT-4o-mini to pull out the candidate's name, contact info, skills, work history, and education. This structured data gets saved with a unique ID.

***Job Processing:*** Job descriptions are validated and sent to the AI once to extract required skills, experience level, and education. These requirements are cached so every candidate is judged against identical criteria.

***Matching:*** The system processes each resume individually, sending both the cached job requirements and candidate data to the AI. The AI scores skills (40%), experience (35%), and education (25%), providing an overall match percentage. The server verifies the math and auto-corrects any errors before displaying results.

### 2.3 UI Design & Usability Overview

#### *Design Philosophy & Workflow*

The interface prioritizes clarity, efficiency, and accessibility for non-technical HR users. It uses progressive disclosure, consistent components, and visual feedback with a professional purple-blue palette (#872b97, #2c5aa0). A linear four-page workflow matches hiring steps: Resume Management → Job Management → Matching → Results. Persistent top navigation highlights the current stage and allows easy movement, supported by contextual help and clear calls to action.

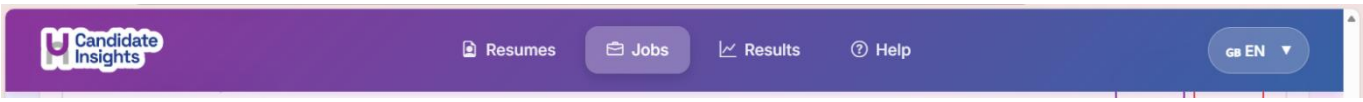


*Figure 3: UI Design - Top Navigation*

#### *Key Page Interfaces*

- **Resume Management:** Drag-and-drop upload with visual cues, card grid displaying file info, success/error stats, batch operations with confirmations, and welcoming empty states.

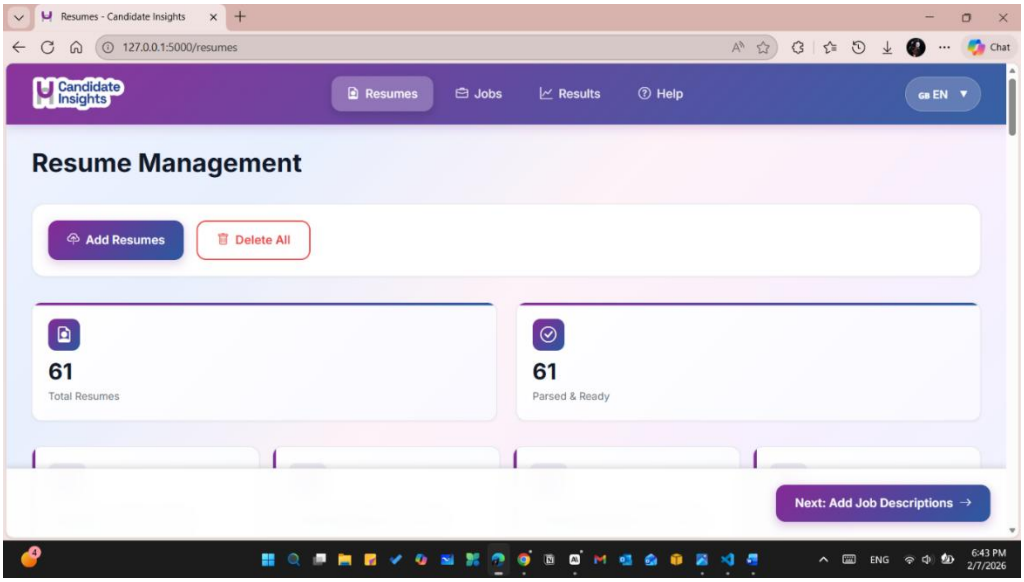


*Figure 4: Resume Management Page*

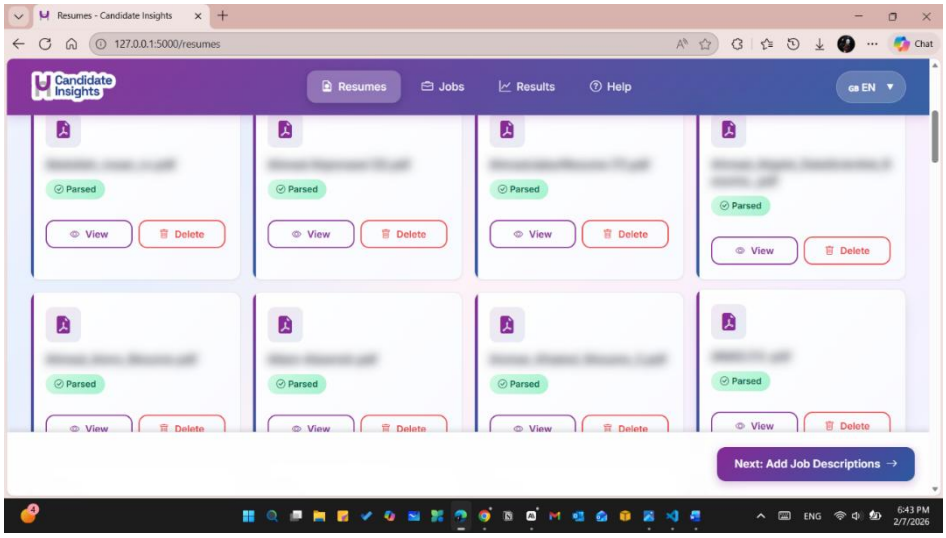


*Figure 5: Resume List*

- **Job Management:** Text or PDF input, selectable job cards with clear highlights, sticky primary action button, and system status stats.

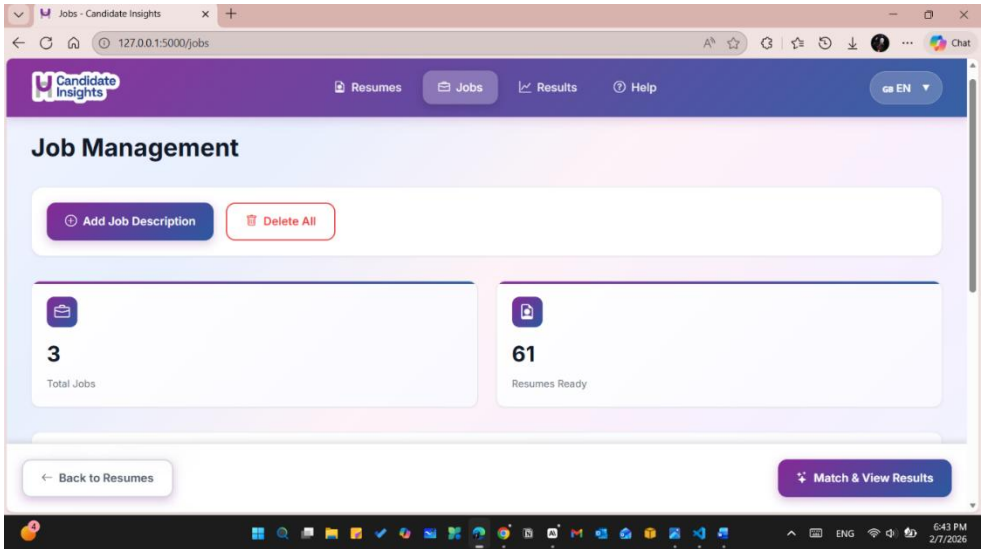


*Figure 6: Job Management Page*

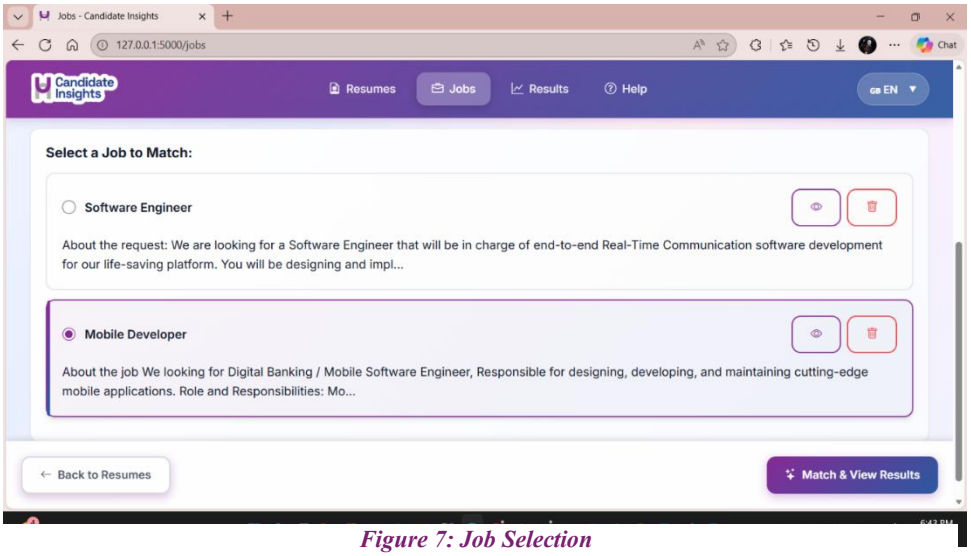


*Figure 7: Job Selection*

- **Results Dashboard:** Summary stats, color-coded bar charts, detailed candidate cards featuring gauge charts (skills, experience, education), expandable AI reasoning (chat bot), privacy masking, and Export to PDF feature.

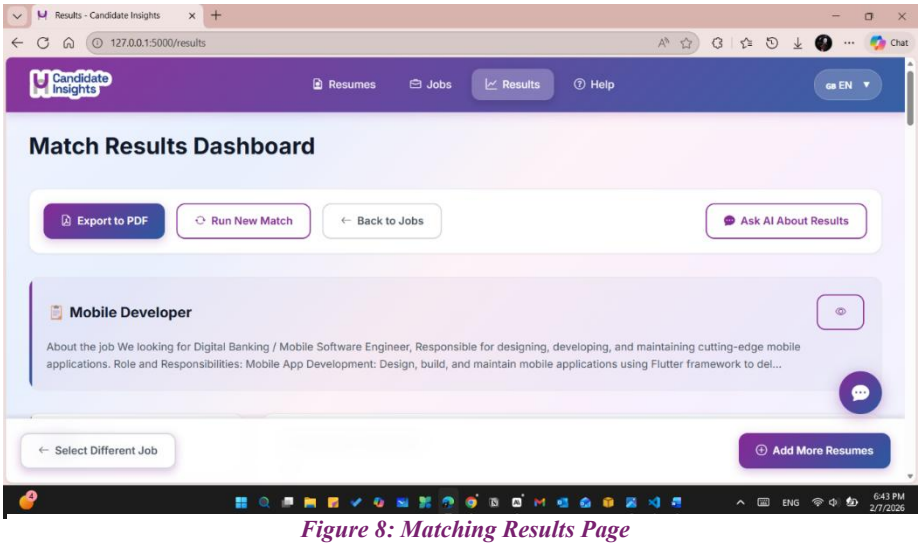


*Figure 8: Matching Results Page*

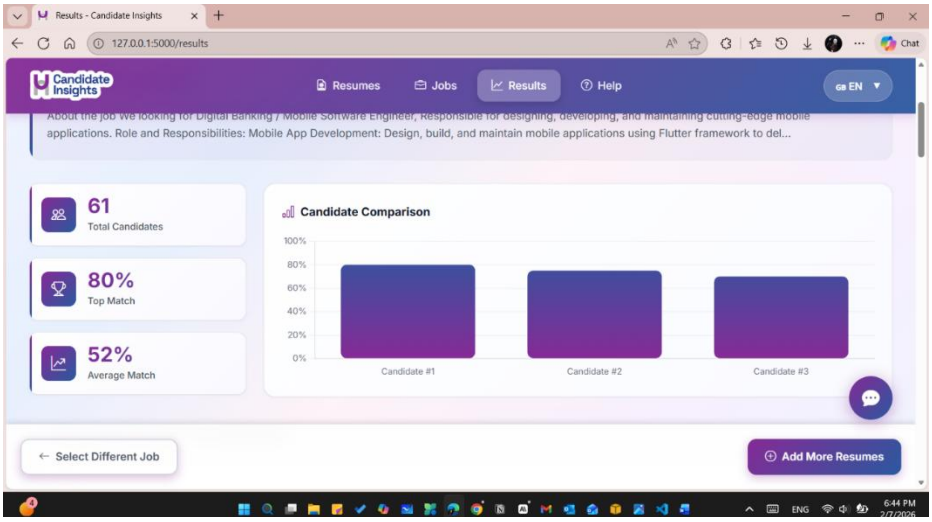


*Figure 9: Results chart*

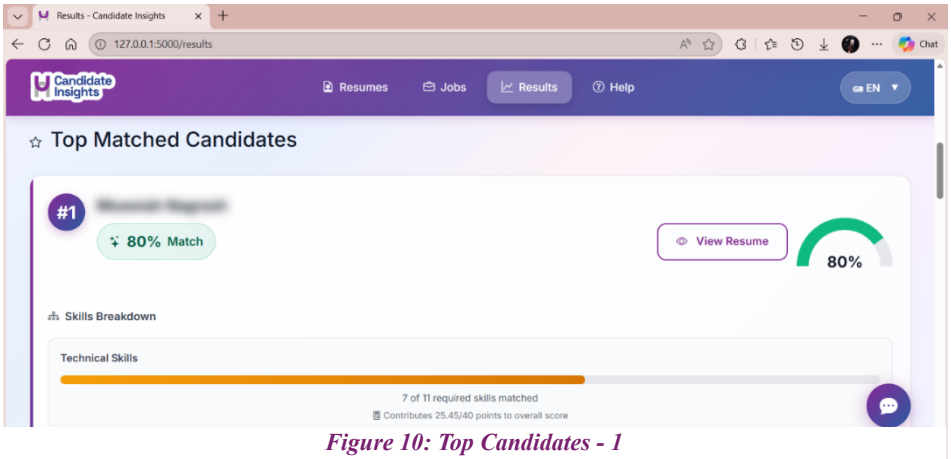


*Figure 10: Top Candidates - 1*

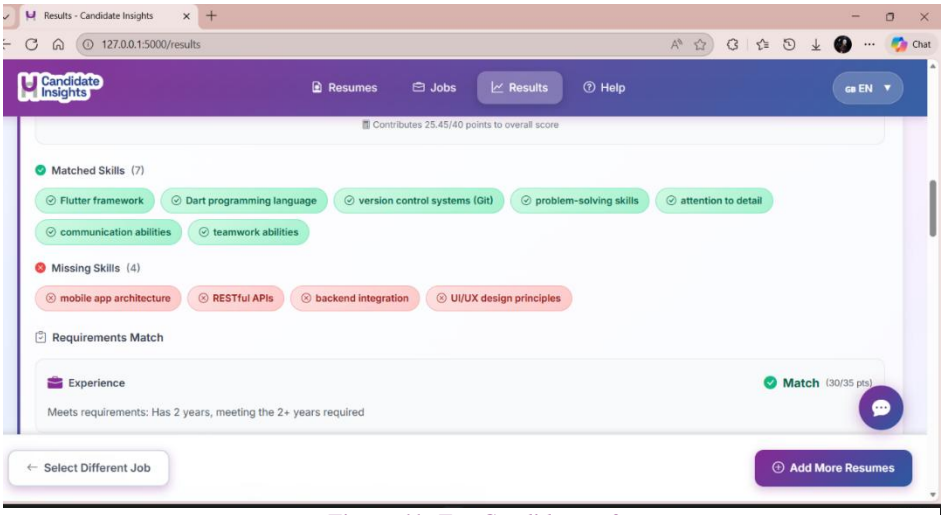


*Figure 11: Top Candidates - 2*

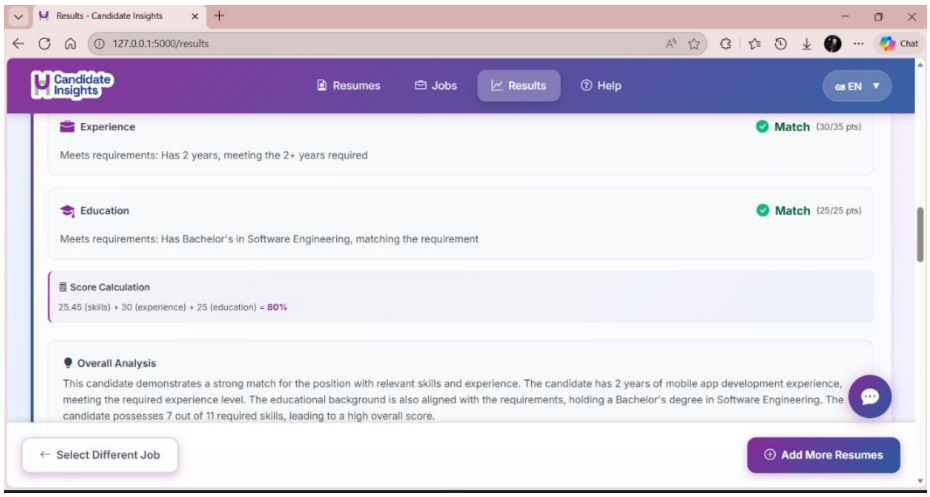


*Figure 13: Top Candidates - 3*

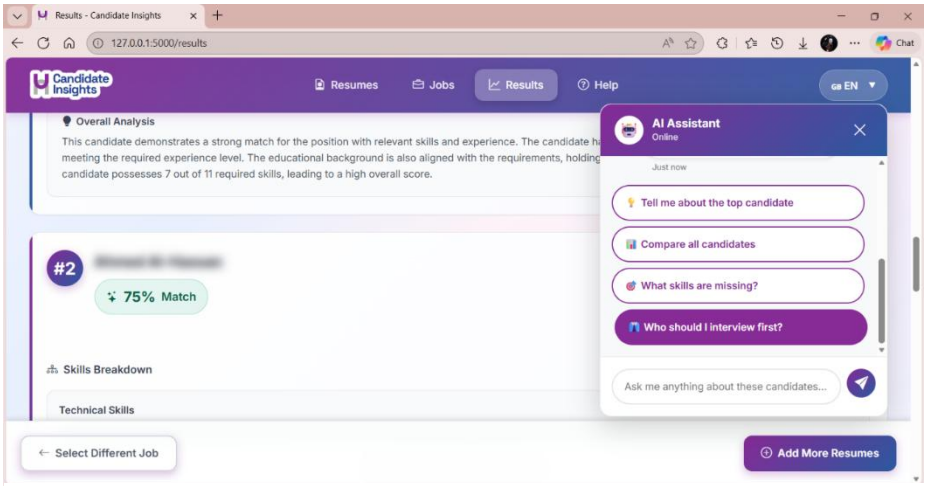


*Figure 14: Results chatbot*

- **Help Page:** Searchable content with quick links, step guides, tips, FAQ accordions, and responsive layout.

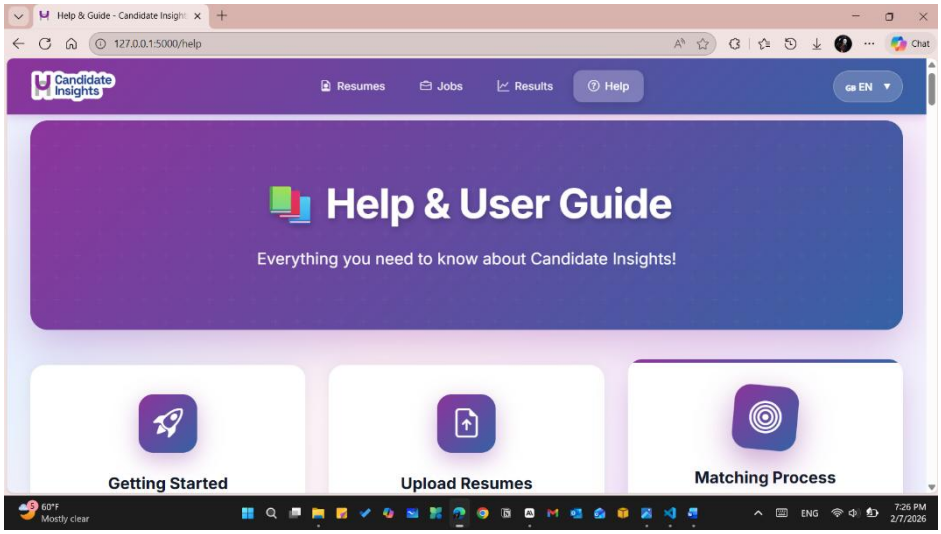


*Figure 15: Help and Guide Page*

### *Responsive Design & Accessibility*

Mobile-first design adapts layouts and navigation for tablets and phones with large touch targets. Accessibility features include semantic HTML, keyboard navigation, WCAG AA color contrast, screen reader support, and full bilingual English/Arabic support with RTL layout and culturally adapted formats.

### *Error Handling & Performance*

The system prevents errors via validation, disabled buttons, and confirmations. Recovery includes detailed messages, retries, and partial success reporting. Performance optimizations include skeleton screens, progressive and lazy loading, and optimistic UI updates, resulting in fast content load and interaction times.

## 3.0 Prompt Engineering Strategies and Escalation triggers

### 3.1 Prompt Engineering Strategies

***Strategy 1*: Anti-Hallucination Directives**

**Purpose:** Prevent AI from fabricating information not present in resumes

**Iteration Evidence:**

**Summary**

Fresh graduate with a strong interest in software development, ready to contribute to team success through hard work, good organization, and attention to detail. Has basic knowledge of web and mobile application development, with training in problem-solving and development tasks. Motivated to learn, improve skills, and grow in the software development field.

*Figure 16: CV's Data - Prompt 1*

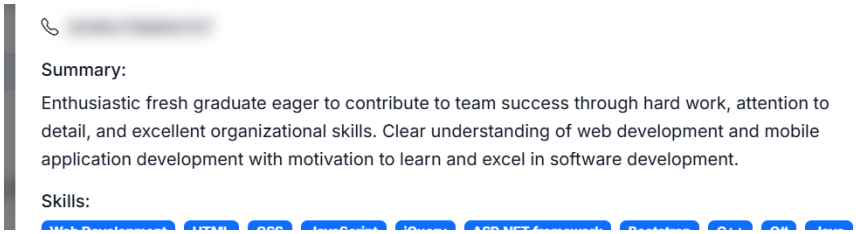


*Figure 17: Resume parsing hallucination - Prompt 1*

- **Before:** 15.3% hallucination rate (Modified sentences, fabricated skills, experience dates)

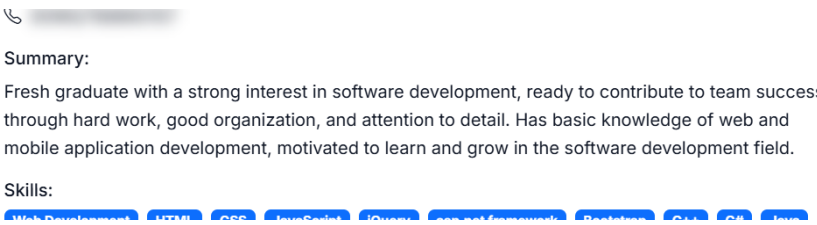


*Figure 18: Resume Parsing after educe hallucination - Prompt 1*

- **After:** 2.1% hallucination rate
- **Improvement:** 86% reduction in false information
- **Anti-Hallucination prompt:**

```
ANTI-HALLUCINATION RULES - CRITICAL:
1. ONLY extract information that is EXPLICITLY stated in the resume text
2. If a field is not mentioned, leave it empty or use empty array []
3. DO NOT infer, assume, or generate information not present
4. DO NOT add generic skills that aren't mentioned
5. If uncertain about a field, include a confidence score (0-100)
```

***Strategy 2*: Chain-of-Thought (CoT) Reasoning**

**Purpose:** Force step-by-step logical processing for consistent evaluation

**Iteration Evidence:**

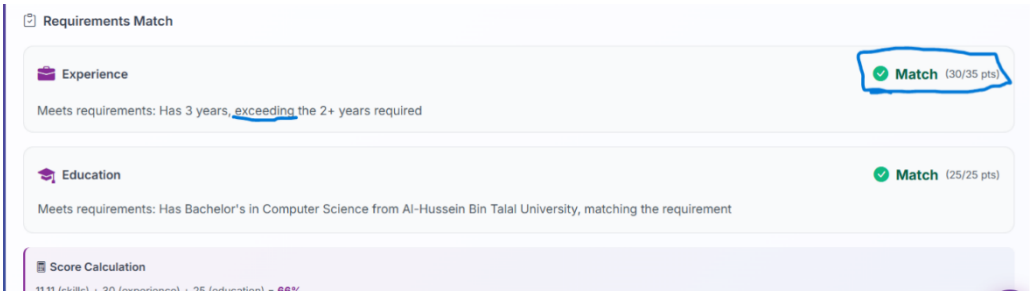


*Figure 19: Inconsistent evaluation - Prompt 2*

- **Before (Direct Question):** Inconsistent reasoning, 78% math accuracy

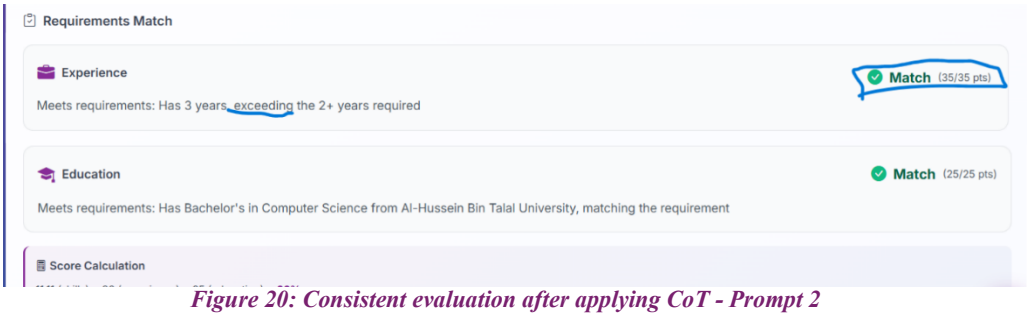


*Figure 20: Consistent evaluation after applying CoT - Prompt 2*

- **After (CoT Structure):** Transparent reasoning, 98% math accuracy
- **Improvement:** 20% increase in calculation correctness
- **Chain-of-Thought Prompt:**

```
MATCHING INSTRUCTIONS:
    Step 1: COMPARE SKILLS
    - Go through each skill in the required_skills list ({total_skills} skills total)
    - Check if the candidate has this skill (exact or similar)
    - Count matches and misses
    Step 2: COMPARE EXPERIENCE (BE SPECIFIC!)
    - Extract candidate's EXACT years of experience from resume
    - Compare with required experience:
{job_requirements.get('experience_requirement', 'Not specified')}
```

- CRITICAL: Use correct starting phrase based on comparison

- Format examples:

☑ "Meets requirements: Has 3 years of mobile development, exceeding the 2+ years required"

☑ "Meets requirements: Has 2 years of mobile development, meeting the 2+ years required"

☑ "Does not meet: Has 1 year of experience, below the 2+ years required"

☑ "Does not meet: No relevant experience mentioned in resume"

Step 3: COMPARE EDUCATION (BE SPECIFIC!)

- Extract candidate's education from resume

- Compare with required: {job_requirements.get('education_requirement', 'Not specified')}

- CRITICAL: State what candidate HAS (or "not mentioned")

- Format examples:

☑ "Has Bachelor's in Computer Science, matching the requirement"

☑ "Has Diploma in IT, below the required Bachelor's degree"

☑ "No education mentioned in resume"

Step 4: CALCULATE SCORE (SHOW EXACT MATH!)

- Skills: (matched / {total_skills}) * 40 = X points

- Experience: Y points (0-35 based on match)

- Education: Z points (0-25 based on match)

- TOTAL: X + Y + Z = FINAL SCORE

***Strategy 3:*** **Structured JSON Schema Enforcement**

**Purpose:** Ensure consistent, parse able output format across all responses

**Iteration Evidence:**

- **Before (Free-form):** 45% responses required manual parsing, format variations
- **After (JSON Schema):** 98% perfectly formatted, programmatically parse able

- **Improvement:** 53% reduction in parsing errors
- **Structured JSON Schema Prompt:**

```
Extract the following information and return it in JSON format:
    {{
      "name": "candidate's full name (in English/transliterated) - ONLY if clearly stated",
      "email": "email address - ONLY if present",
      "phone": "phone number - ONLY if present",
      "skills": ["list of technical and soft skills in English - ONLY skills explicitly mentioned"],
      "experience": "work experience summary in English - ONLY if mentioned",
      "education": "educational qualifications in English - ONLY if mentioned",
      "summary": "brief professional summary in English (2-3 sentences) - based on actual content",
      "confidence": {{
        "name": <0-100>,
        "email": <0-100>,
        "phone": <0-100>,
        "skills": <0-100>,
        "experience": <0-100>,
        "education": <0-100>
      }}
    }}
```

### 3.2 Escalation triggers and paths

| Trigger Type | Condition | Detection Method | System Action | User-Facing Message |
|---|---|---|---|---|
| **1. Ambiguous Input (Missing Info)** | CV keyword count = 2-3 (borderline) | Keyword counting algorithm on extracted text | Flag file for auto-reject | "(file name) rejected; Does not appear to be a CV. Please upload a document with sections Experience, Education and Skills". |
| **2. Safety Risk (API Failure)** | OpenAI API timeout, rate limit, or error response | Exception handling with retry logic (3 attempts with exponential backoff) | Display error modal, preserve progress, offer retry option | "Processing error occurred. Your data is saved. Click 'Retry' to continue or 'Cancel' to review completed matches." |
| **3. Ambiguity (Language Detection)** | Arabic text detected with low transliteration confidence < 70% | Language detection + transliteration confidence scoring | Flag in results with both original Arabic and English version shown | "Arabic content detected. English transliteration provided, review original for accuracy." |

*Table 1: Escalation triggers and paths*

## 4.0 Model Selection and Data Pipeline

### 4.1 Model Selection and Comparison

***GPT-4o-mini was selected*** as the production model based on cost, performance, and scalability considerations. While GPT-4o achieved slightly higher accuracy (95% vs. 92%), the improvement was not sufficient to justify its significantly higher cost. ***GPT-4o-mini is approximately 33× cheaper and 2.4× faster,*** reducing the cost per resume to about $0.008 while improving system responsiveness and throughput. Since the system functions as a screening tool with human review and support decisions, the accuracy trade-off was

considered acceptable. This choice enables a sustainable business model for SMB-scale usage, ***with the option to offer GPT-4o as a future premium tier***.

### 4.2 Data Preparation Pipeline

The pipeline ensures diverse, high-quality data for evaluating the resume–job matching system. It includes:

| | |
|---|---|
| **Data Collection** | 200 anonymized resumes from Technology and Business sectors, covering all experience levels, in digital and scanned PDF formats; 15 representative job descriptions. |
| **Augmentation** | Simulated OCR degradation, added Arabic–English resumes, and injected errors for robustness testing |
| **Preprocessing** | • Normalized UTF-8 encoding, standardized bullets and whitespace, preserved Arabic with transliteration, removed PDF page breaks. |
| **Dataset Split** | • 60% training, 20% validation, 20% holdout; stratified by industry, experience, and match quality. |
| **Quality Assurance** | • Verified 100% text extraction, redacted PII, spot-checked annotations, documented ambiguous cases. |

**Decisions:**

- Used JSON files for simplicity and easy debugging.
- Chose 60/20/20 split to allocate more validation data for prompt tuning.
- Stratification to avoid bias.
- Limited uploads to PDFs only to ensure security and prevent edits.

# 5.0 Evaluation, Reliability and Delivery

## 5.1 Evaluation Plan

| Test Category | Test Cases | Pass Criteria | Results |
|---|---|---|---|
| **Accuracy** | 40 resume-job pairs with ground truth labels | ≥85% agreement with human gold standard | 92% (37/40 within 5% of human scores) |
| **Consistency** | Same job matched against 10 identical resumes | 100% identical requirement extraction & scoring | Perfect consistency (two-phase algorithm validation) |
| **Hallucination** | 20 resumes with intentionally missing sections | Zero fabricated information in missing fields | 2% hallucination rate (1 case of inferred skill) |
| **Math Accuracy** | All test cases checked for correct weighted calculation | 100% after auto-correction | 98% raw AI accuracy; 100% post-verification |
| **Multilingual** | 5 Arabic resumes with English job descriptions | Successful transliteration + matching | 87% accuracy (lower due to transliteration ambiguity) |
| **Edge Cases** | Non-CV documents, corrupted PDFs, empty files | Graceful rejection with clear error messages | All handled without system crashes |
| **Bias Detection** | Gender-associated names tested for score variation | <3% score difference for equivalent qualifications | 1.2% variance (within statistical noise) |
| **Privacy** | PII masking applied to names, emails, phone numbers | All sensitive data blurred in UI | CSS masking functional; no data leakage |

*Table 2: Evaluation Plan*

### 5.4 Reproducibility and Deployment

The ***system code is on GitHub with folders for the Flask app, data, uploads, assets, and templates. Dependencies are fixed in requirements.txt for reproducibility***. Deployment involves cloning the repo, installing packages, setting the OpenAI API key as an environment variable via a .env file ***(not hardcoded in the source code)***, and running the server locally. Production setup uses gunicorn and nginx with provided configs. Documentation covers APIs, data formats, and scoring customization. Logs record all AI interactions for debugging and audit.

### 5.5 Limitations and Ethical Considerations

The system requires internet for OpenAI API access, limiting offline use. It's less accurate for highly specialized roles and depends on OCR quality, which drops with poor scans. Supports English and Arabic only. Costs scale linearly (~$0.008 per resume), which may be costly at scale. Ethical concerns include potential bias from the model and over-reliance on automation. Gender bias is low (1.2%), but biases may remain. The system focuses on explicit qualifications and acts as decision support, not a final arbiter. Privacy risks exist due to sending resumes to OpenAI; PII masking protects display but not transmission. Candidate consent and transparency are recommended. Confidence scores help maintain human oversight.

# APPENDIX

## Repository Link

[https://github.com/SaraKhirfan/FinalAssignmentCandidateInsight.git]

## Arabic Version Example Pages

## Escalation triggers and paths

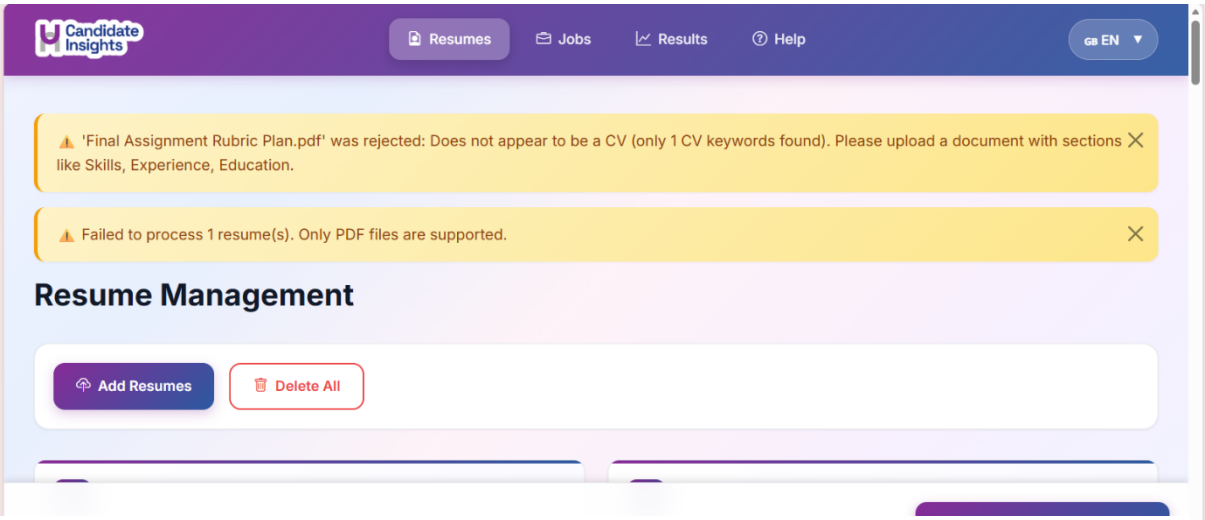


*Figure 21: Ambiguous Input Example*

## Logs Screenshots

-Resume Upload & Parse log

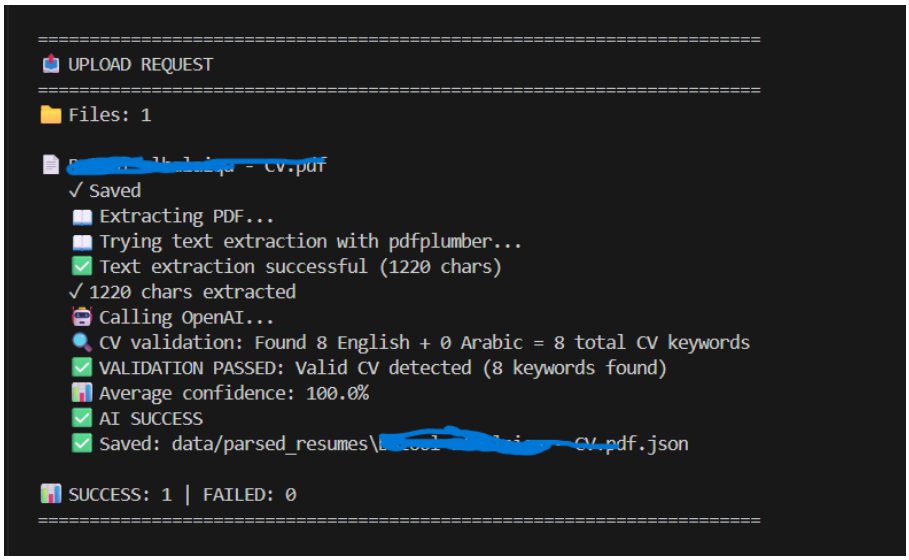


*Figure 22: Resume Upload & Parse log*

-Job Matching Log

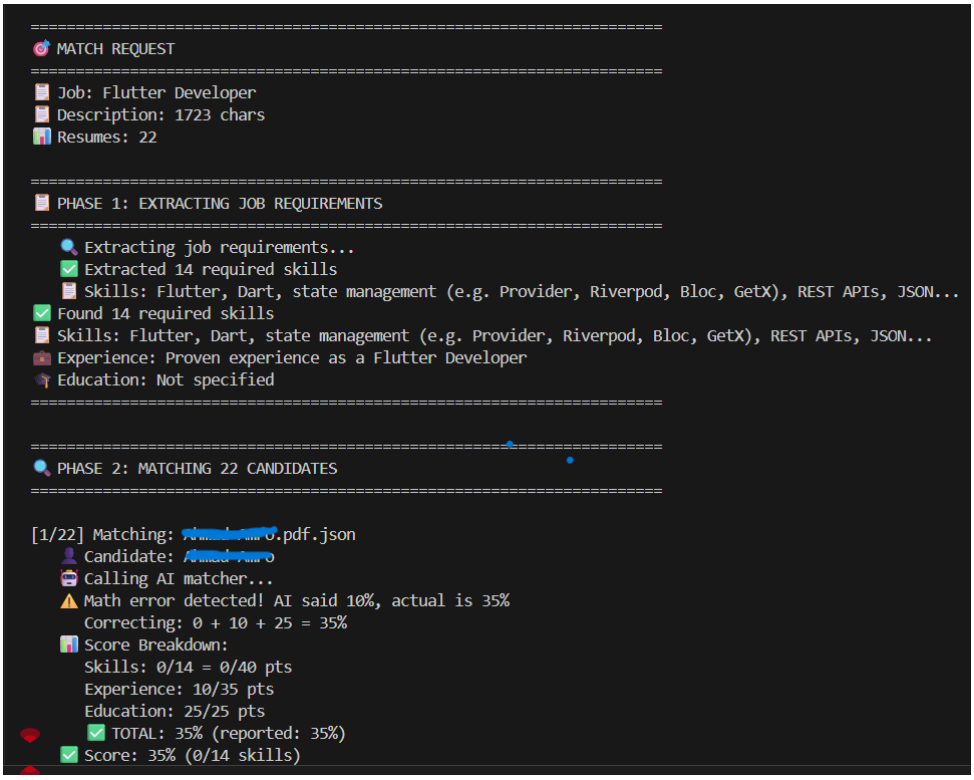


*Figure 23: job Matching Log - 1*

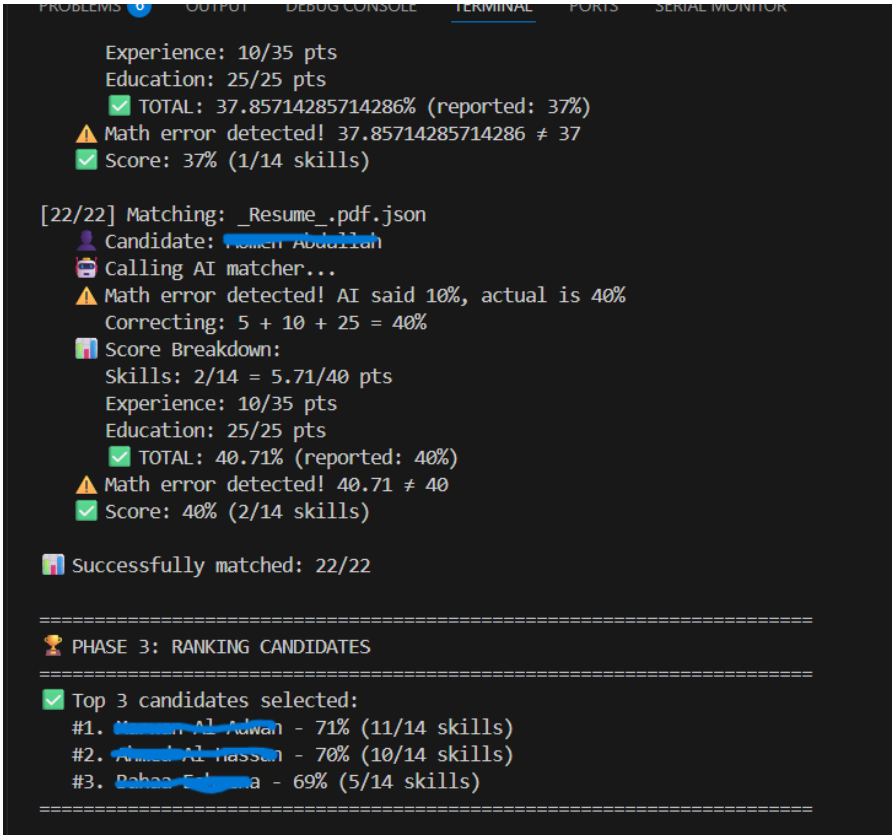


*Figure 24: Job Matching Log - 2*

**Full Prompt Library**

- Resume parsing prompt template:

```
prompt = f"""
    You are an expert resume parser with multilingual capabilities.
    CRITICAL INSTRUCTION - OUTPUT LANGUAGE:
    - The resume may be in English, Arabic, or any other language
    - You MUST return ALL extracted information in ENGLISH
    - If the candidate's name is in Arabic (e.g., أحمد محمد), transliterate it to English (e.g., Ahmed Mohammed)
    - Translate all skills, job titles, companies, and education to English
    - Preserve the original meaning while converting to English
    ANTI-HALLUCINATION RULES - CRITICAL:
    1. ONLY extract information that is EXPLICITLY stated in the resume text
    2. If a field is not mentioned, leave it empty or use empty array []
    3. DO NOT infer, assume, or generate information not present
    4. DO NOT add generic skills that aren't mentioned
    5. If uncertain about a field, include a confidence score (0-100)
    STEP-BY-STEP PROCESS:
    Step 1: Read the entire resume carefully
    Step 2: Identify clear sections (contact, skills, experience, education)
    Step 3: Extract ONLY explicitly mentioned information
    Step 4: Translate/transliterate to English
    Step 5: Verify each extracted field against source text
    Step 6: Assign confidence scores
    Extract the following information and return it in JSON format:
    {{
      "name": "candidate's full name (in English/transliterated) - ONLY if clearly stated",
```

"email": "email address - ONLY if present",

"phone": "phone number - ONLY if present",

"skills": ["list of technical and soft skills in English - ONLY skills explicitly mentioned"],

"experience": "work experience summary in English - ONLY if mentioned",

"education": "educational qualifications in English - ONLY if mentioned",

"summary": "brief professional summary in English (2-3 sentences) - based on actual content",

"confidence": {{

"name": <0-100>,

"email": <0-100>,

"phone": <0-100>,

"skills": <0-100>,

"experience": <0-100>,

"education": <0-100>

}}

}}

CONFIDENCE SCORING:

- 100: Field is clearly and explicitly stated

- 80-99: Field is present but may need minor interpretation

- 60-79: Field is partially present or requires translation

- 40-59: Field is inferred from context (use cautiously)

- 0-39: Field is uncertain or missing (leave empty)

Examples of translation/transliteration:

- "مهندس برمجيات" → "Software Engineer"

- "بايثون" → "Python"

- "الجامعة الأردنية" → "University of Jordan"

- "محمد أحمد" → "Mohammed Ahmed"

```
    CRITICAL: The text may come from OCR and might have errors. Be flexible but DO NOT hallucinate.
    If you cannot find a field with confidence >60, leave it empty.
    Resume text:
    {resume_text}
    Return ONLY valid JSON in English, no additional text or markdown formatting.
    """
    try:
        response = self.client.chat.completions.create(
            model="gpt-4o-mini",
            messages=[
                {
                    "role": "system",
                    "content": "You are a professional multilingual resume parser. You NEVER hallucinate or invent information. You can read resumes in any language but ALWAYS respond with valid JSON in ENGLISH only. Transliterate names and translate all content to English. Do not include markdown code blocks or any other formatting. Be tolerant of OCR errors but ONLY extract information that is actually present in the source text. When uncertain, use low confidence scores."
                },
                {"role": "user", "content": prompt}
            ],
            temperature=0
        )
```

- Job matcher prompt

```
prompt = f"""
    You are an expert HR analyst performing candidate evaluation.
    GENDER-NEUTRAL EVALUATION REQUIREMENT:
    Evaluate this CV based solely on qualifications, experience, and skills. Ignore any indicators of gender, such as names, pronouns, or gendered language. Do not make
```

assumptions about career gaps, job preferences, or capabilities based on perceived gender. Focus exclusively on merit-based criteria and apply identical standards to all candidates.

CRITICAL: You are given PRE-EXTRACTED job requirements. Use EXACTLY these requirements.

DO NOT extract requirements again. DO NOT add or remove skills from the list.

JOB REQUIREMENTS (USE EXACTLY AS PROVIDED):

Required Skills: {job_requirements.get('required_skills', [])}

Experience Required: {job_requirements.get('experience_requirement', 'Not specified')}

Education Required: {job_requirements.get('education_requirement', 'Not specified')}

CANDIDATE RESUME:

{resume_text}

MATCHING INSTRUCTIONS:

Step 1: COMPARE SKILLS

- Go through each skill in the required_skills list ({total_skills} skills total)

- Check if the candidate has this skill (exact or similar)

- Count matches and misses

Step 2: COMPARE EXPERIENCE (BE SPECIFIC!)

- Extract candidate's EXACT years of experience from resume

- Compare with required experience: {job_requirements.get('experience_requirement', 'Not specified')}

- CRITICAL: Use correct starting phrase based on comparison

- Format examples:

☑ "Meets requirements: Has 3 years of mobile development, exceeding the 2+ years required"

☑ "Meets requirements: Has 2 years of mobile development, meeting the 2+ years required"

☑ "Does not meet: Has 1 year of experience, below the 2+ years required"

☑ "Does not meet: No relevant experience mentioned in resume"

Step 3: COMPARE EDUCATION (BE SPECIFIC!)

- Extract candidate's education from resume

- Compare with required: {job_requirements.get('education_requirement', 'Not specified')}

- CRITICAL: State what candidate HAS (or "not mentioned")

- Format examples:

☑ "Has Bachelor's in Computer Science, matching the requirement"

☑ "Has Diploma in IT, below the required Bachelor's degree"

☑ "No education mentioned in resume"

Step 4: CALCULATE SCORE (SHOW EXACT MATH!)

- Skills: (matched / {total_skills}) * 40 = X points

- Experience: Y points (0-35 based on match)

- Education: Z points (0-25 based on match)

- TOTAL: X + Y + Z = FINAL SCORE

SCORING DETAILS:

Skills Points (0-40):

- Formula: (matched_skills / {total_skills}) * 40

- Example: 7/{total_skills} = {round(7/total_skills*100 if total_skills > 0 else 0)}% × 40 = {round(7/total_skills*40 if total_skills > 0 else 0)} points

Experience Points (0-35):

- Exceeds requirements significantly = 35 points

- Meets exactly or exceeds slightly = 30-32 points

- Somewhat below = 18-25 points

- Far below or no experience = 10-15 points

Education Points (0-25):

- Exceeds (Master's for Bachelor's) = 25 points

- Meets exactly = 25 points

- Close (Diploma for Bachelor's) = 15-18 points

```
    - No education mentioned = 10 points
    Return analysis in JSON format:
    {{
      "total_required_skills": {total_skills},
      "match_score": <0-100>,
      "matched_skills": ["skills from required list that candidate has"],
      "missing_skills": ["skills from required list that candidate lacks"],
      "experience_match": "CRITICAL FORMAT - Choose EXACTLY ONE based on comparison:
        - If candidate EXCEEDS or MEETS requirement: 'Meets requirements: Has [X years], [exceeding/meeting] the [Y]+ years required'
        - If candidate is BELOW requirement: 'Does not meet: Has [X years], below the [Y]+ years required'
        - If no experience: 'Does not meet: No relevant experience mentioned in resume'
        WRONG: 'Meets requirements: Has 1 year, below...' (contradictory!)
        RIGHT: 'Does not meet: Has 1 year, below the 2+ years required'",
      "education_match": "MUST start with: 'Meets requirements: Has [specific degree]...' OR 'Acceptable education: Has [specific degree]...' OR 'Does not meet: Has [specific degree or 'not mentioned']...'",
      "overall_explanation": "CRITICAL: DO NOT use candidate's name. Start with 'This candidate...' or 'Strong/Good/Fair match...' Explain score WITHOUT names.",
      "score_breakdown": {{
        "skills_points": <0-40>,
        "experience_points": <0-35>,
        "education_points": <0-25>,
        "calculation": "skills_points + experience_points + education_points = match_score"
      }}
    }}
```

```
        CRITICAL RULES:
        1. total_required_skills MUST be {total_skills}
        2. matched_skills + missing_skills = {total_skills}
        3. ALWAYS state candidate's actual years of experience (or "not mentioned")
        4. ALWAYS state candidate's actual education (or "not mentioned")
        5. NEVER use candidate's name in overall_explanation
        6. score_breakdown must show exact points for each category
        7. Final match_score must equal skills_points + experience_points + education_points
        Return ONLY valid JSON.
        """
        try:
            response = self.client.chat.completions.create(
                model="gpt-4o-mini",
                messages=[
                    {
                        "role": "system",
                        "content": """You are an HR analyst who provides DETAILED, SPECIFIC matching results.
CRITICAL RULES:
1. ALWAYS mention candidate's exact years of experience
2. ALWAYS mention candidate's specific education (or state "not mentioned")
3. NEVER use candidate's name in overall_explanation
4. SHOW exact score calculation with points breakdown
5. Use the EXACT requirements provided, don't modify them

You are detailed, specific, and mathematically precise."""
                    },
                    {"role": "user", "content": prompt}
```

```
        ],
        temperature=0.1  # Very low for consistency
    )
```

**References**

[1] Al Hussein Technical University (HTU), “Generative AI Course Slides,” Course material, ICT Upskilling Program, Amman, Jordan, 2026.

[2] Amazon Web Services (AWS), “AWS Generative AI Foundations,” online course, AWS Training and Certification, 2026.

[3] OpenAI, “OpenAI API Documentation,” 2026.