talentz.ai

Vibe Recruiting: AI Resume Screening Workflow

A Vibe Recruiting Playbook by talentz.ai

Series: Tech Vibes with Shaphy

Version 1.0 | Last Updated: October 2025

Goal

This guide provides a complete, step-by-step setup for building an automated resume evaluation system using n8n and OpenAI. In under two hours, you will create a workflow that receives resumes, analyzes them, and ranks candidates objectively based on job requirements.

Prerequisites

Required Accounts & Access

1. n8n Account
   - Cloud account (https://n8n.io)

OR

   - Self-hosted instance (version 1.0+)

2. OpenAI Account
   - Active API key with GPT-4 access
   - Minimum $5 credit balance recommended
   - API key: https://platform.openai.com/api-keys

3. Basic Knowledge
   - JSON syntax fundamentals
   - Basic understanding of API concepts
   - Familiarity with workflow automation (helpful but not required)

Estimated Costs

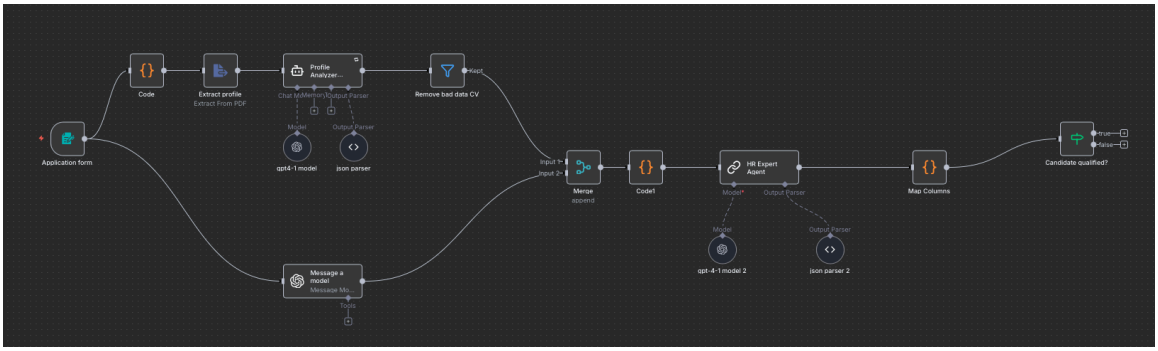
- Setup: Free
- Per Resume: $0.03 - $0.10 (OpenAI API usage)
- n8n Cloud (optional): $20/month for starter plan
- Self-hosting (optional): $5-50/month depending on provider

Time Investment

- Initial Build: 30-45 minutes
- Customization: 15-30 minutes
- Testing: 15 minutes
- Total: ~1-2 hours

Workflow Overview

Workflow Architecture



Data Flow Summary

1. Candidate submits resume + job info via form
2. PDF is extracted to text
3. AI structures raw text into JSON profile
4. Job description is enhanced for clarity
5. Profiles and JD are merged
6. AI evaluates each candidate
7. System scores and filters results
8. Top candidates are routed automatically

Step-by-Step Setup

Part 1: Setting Up Your n8n Workspace

Step 1.1: Create New Workflow

1. Log into your n8n instance
2. Click "New Workflow" button (top right)
3. Name it: "AI Resume Screening System"
4. Click into the canvas

Step 1.2: Configure Credentials

Add OpenAI Credentials:

1. Click Settings (gear icon in top right)
2. Select Credentials tab
3. Click "Add Credential" button
4. Search for "OpenAI"
5. Fill in:
   - Credential Name: "OpenAI account"
   - API Key: Your OpenAI API key
6. Click "Save"

Part 2: Building the Form Trigger

Step 2.1: Add Form Trigger Node

1. Click the "+" button on the canvas
2. Search for "Form Trigger"
3. Select "Form Trigger" node

Step 2.2: Configure Form Settings

Basic Settings:

1. Click on the Form Trigger node
2. In the parameters panel:
   - Form Title: "Resume Review Form"
   - Form Description: "Upload all applicant resumes below and select the corresponding job positions. Our SmartHR Agent will review each submission and provide a detailed evaluation for your consideration."

Form Fields:

Click "Add Field" twice to create two fields:

Field 1 - Resume Upload:
- Field Label: "CV"
- Field Type: "File"
- Accept File Types: ".pdf"
- Required Field: Toggle ON

Field 2 - Job Description:
- Field Label: "Job Description"
- Field Type: "Text" (default)
- Required Field: Toggle ON

Step 2.3: Test Form

1. Click "Test workflow" button
2. Click on the Form Trigger node
3. Click "Open Form" link
4. Verify the form displays correctly
5. Return to n8n canvas

Part 3: Processing Resume Files

Step 3.1: Add Code Node for Binary Processing

1. Click the "+" after Form Trigger
2. Search for "Code"
3. Select "Code" node
4. Rename to: "Code" (or keep default)

Configure Code:

```
const data = $input.item.json;
const binaryData = $input.item.binary;

let output = [];

Object.keys(binaryData)
  .filter(label => label.startsWith("CV_"))
  .forEach(label => {
    output.push({
      json: data,
      binary: { data: binaryData[label] }
    });
```

```
  });

return output;
```

Explanation: This code extracts all uploaded CV files from the form submission and prepares them for processing.

Step 3.2: Add Extract From File Node

1. Click "+" after the Code node
2. Search for "Extract from File"
3. Select the node

Configure:
- Operation: "Extract text from PDF"
- Binary Property: "data"
- All other settings: Keep defaults

What it does: Converts PDF resumes into plain text while preserving structure.

Part 4: Setting Up AI Profile Analysis

Step 4.1: Add Profile Analyzer Agent

1. Click "+" after Extract Profile
2. Search for "Agent"
3. Select "AI Agent" node
4. Rename to: "Profile Analyzer Agent"

Configure Agent:
1. In Prompt Type: Select "Define below"
2. In Prompt field:

```
Please extract all relevant information from this candidate:
CV Content:
{{ $json["text"] }}
```

3. Toggle "Require Specific Output Format" ON

Step 4.2: Add Language Model (GPT-4.1-mini)

1. Below the Profile Analyzer Agent, click "+" on "Language Model" connection
2. Search for "OpenAI Chat Model"
3. Select it and rename to: "gpt4-1 model"

Configure:
- Credentials: Select "OpenAI account" (created earlier)

- Model: Select "gpt-4.1-mini" from dropdown
- Leave other settings default

Step 4.3: Add JSON Output Parser

1. Below Profile Analyzer Agent, click "+" on "Output Parser" connection
2. Search for "Structured Output Parser"
3. Select it and rename to: "json parser"

Configure JSON Schema:

Paste this schema in the "JSON Schema Example" field:

```
{
  "full_name": "",
  "job_title": "",
  "summary": "",
  "contact": {
    "email": "",
    "phone": "",
    "address": "",
    "linkedin": "",
    "website": ""
  },
  "education": [
    {
      "degree": "",
      "field_of_study": "",
      "institution": "",
      "location": "",
      "start_year": "",
      "end_year": ""
    }
  ],
  "certifications": [
    {
      "name": "",
      "issuer": "",
      "date_obtained": "",
      "expiration_date": ""
    }
  ],
  "work_experience": [
    {
      "company": "",
```

```
      "position": "",
      "location": "",
      "start_date": "",
      "end_date": "",
      "responsibilities": [""],
      "achievements": [""]
    }
  ],
  "skills": {
    "technical_skills": [""],
    "soft_skills": [""],
    "languages": [
      {
        "language": "",
        "proficiency": ""
      }
    ],
    "tools_and_technologies": [""]
  },
  "projects": [
    {
      "name": "",
      "description": "",
      "technologies_used": [""],
      "role": "",
      "duration": ""
    }
  ],
  "awards": [
    {
      "title": "",
      "issuer": "",
      "date": "",
      "description": ""
    }
  ],
  "volunteer_experience": [
    {
      "organization": "",
      "role": "",
      "description": "",
```

```
      "start_date": "",
      "end_date": ""
    }
  ],
  "additional_information": {
    "availability": "",
    "preferred_location": "",
    "salary_expectation": "",
    "work_authorization": "",
    "other_notes": ""
  }
}
```

Save all nodes and test this section.

Part 5: Job Description Enhancement

Step 5.1: Add Message a Model Node

1. Go back to the Form Trigger node
2. Add a parallel branch by clicking "+"
3. Search for "OpenAI"
4. Select "Message a Model" node

Configure:

1. Credentials: Select "OpenAI account"
2. Model: Select "gpt-5-mini"
3. Click "Add Message" twice

Message 1 (System):

- Role: System
- Content:

```
As an expert Recruiting Manager rewrite the JD to ensure its detailed and crisp, do not exaggerate or hallucinate
```

Message 2 (User):

- Role: User
- Content:

```
{{ $json["Job Description"] }}
```

Part 6: Data Quality & Merging

Step 6.1: Add Filter Node

1. After Profile Analyzer Agent, add a Filter node
2. Rename to: "Remove bad data CV"
Configure Conditions:
- Condition: "Full Name" is not empty
- Expression: `={{ $json.output.full_name }}`
- Operation: "Is Not Empty"

Step 6.2: Add Merge Node

1. Add a Merge node to the canvas
2. Connect "Remove bad data CV" output to Input 1
3. Connect "Message a model" output to Input 2
Configure:
- Mode: "Merge By Index"
- Join: "Wait for all messages"

Step 6.3: Add Code Node for Data Preparation

1. After Merge, add another Code node
2. Rename to: "Code1"
Configure Code:

```
// Raw merged input (first N items are candidate profiles, last item is JD)
const items = $input.all();

// Extract job description (assuming it's the last element)
const jdItem = items[items.length - 1];
const job_description = jdItem.json.text;

// Prepare output array
const results = [];

// Loop through all items except the last one (JD)
for (let i = 0; i < items.length - 1; i++) {
  const candidate = items[i].json.output;
  results.push({
    json: {
      candidate_profile: candidate,
      job_description: job_description,
      applied_position: $('Application form').first().json['Job Role']
    }
```

```
  });
}

return results;
```

Purpose: Combines each candidate profile with the enhanced job description for evaluation.

Part 7: HR Expert Evaluation System

Step 7.1: Add HR Expert Chain Node

1. After Code1, add "LLM Chain" node
2. Rename to: "HR Expert Agent"

Configure Prompt:

1. Prompt Type: "Define below"
2. Prompt field:

```
Evaluate the candidate for the following position ({{ $json.position }}) with job description: {{ $json.job_description }}
----------
Below is the extracted candidate profile:
{{ $json.candidate_profile.toJsonString() }}
```

3. Toggle "Require Specific Output Format" ON
4. Click "Add Option"
5. Select "System Message"
6. System Message content:

```
You are a professional AI hiring assistant tasked with evaluating job candidates based on their structured resume data and a provided job description. Your goal is to assess the candidate's suitability for the role, identifying key matches and gaps in skills, experience, and qualifications.

You will receive:
- `candidate_profile`: a JSON object containing extracted information from the candidate's resume.
- `job_description`: a plain text description of the role.

Your job is to:
1. Parse and understand the key job requirements (e.g., required skills, years of experience, qualifications).
```

```
2. Match the candidate's profile to the job requirements.
3. Provide a structured evaluation with ratings and reasoning.

Always be objective, use professional language, and format your output clearly using markdown.
```

Step 7.2: Add Language Model for Evaluation

1. Below HR Expert Agent, add "OpenAI Chat Model"
2. Rename to: "gpt-4-1 model 2"

Configure:

- Credentials: Select "OpenAI account"
- Model: "gpt-4.1-mini"

Step 7.3: Add Output Parser for Evaluation

1. Add "Structured Output Parser"
2. Rename to: "json parser 2"

JSON Schema:

```
{
  "full_name":"",
  "phone":"",
  "address":"",
  "email":"",
  "applied_position":"",
  "overall_fit_score": 0,
  "recommendation": "",
  "job_match_summary": {
    "required_experience": "",
    "skill_match": "",
    "education_certifications": "",
    "soft_skills_traits": ""
  },
  "strengths": [""],
  "concerns_or_gaps": [""],
  "final_notes": ""
}
```

Part 8: Results Processing & Filtering

Step 8.1: Add Map Columns Node

1. After HR Expert Agent, add Code node

2. Rename to: "Map Columns"
Configure Code:

```
const evaluationResult = item.json.output;

return {
  json: {
    full_name: evaluationResult.full_name || '',
    email: evaluationResult.email || '',
    phone: evaluationResult.phone || '',
    position: evaluationResult.applied_position,
    overall_fit_score: evaluationResult.overall_fit_score,
    recommendation: evaluationResult.recommendation,
    strengths: Array.isArray(evaluationResult.strengths) ?
evaluationResult.strengths.join(', ') : '',
    concerns_or_gaps: Array.isArray(evaluationResult.concerns_or_gaps) ?
evaluationResult.concerns_or_gaps.join(', ') : '',
    final_notes: evaluationResult.final_notes || '',
    submitted_date: $now.toFormat("yyyyLLdd-HHmmss")
  }
};
```

Step 8.2: Add Qualification Filter

1. After Map Columns, add "If" node
2. Rename to: "Candidate qualified?"
Configure Condition:
- Condition 1: "overall_fit_score" >= 8
- Expression: `={{ $('HR Expert Agent').item.json.output.overall_fit_score }}`
- Operation: "Greater Than or Equal"
- Value: 8
Result:
- True branch: Qualified candidates (score 8+)
- False branch: Candidates needing further review

Part 9: Output Configuration (Optional)

You can add nodes after the "Candidate qualified?" node to:
Option A: Send to Google Sheets
1. Add Google Sheets node to True branch
2. Configure to append qualified candidates
Option B: Send Email Notifications
1. Add Send Email node

2. Notify hiring manager of top candidates

Option C: Save to Database

1. Add PostgreSQL/MySQL node
2. Store all evaluations for tracking

Option D: Slack Notification

1. Add Slack node
2. Post qualified candidates to hiring channel

Configuration Details

Customizing the Evaluation Criteria

To adjust what the AI evaluates, modify the HR Expert Agent system message:

For Technical Roles:

Focus heavily on:
- Technical skills proficiency (weight: 40%)
- Project experience (weight: 30%)
- Problem-solving abilities (weight: 20%)
- Team collaboration (weight: 10%)

For Leadership Roles:

Focus heavily on:
- Management experience (weight: 35%)
- Strategic thinking (weight: 25%)
- Team building (weight: 20%)
- Communication skills (weight: 20%)

For Entry-Level Roles:

Focus on:
- Educational background (weight: 30%)
- Relevant coursework/projects (weight: 25%)
- Internship experience (weight: 20%)
- Learning ability and potential (weight: 25%)

Adjusting the Scoring Threshold

Change the score in "Candidate qualified?" node:
- Highly Competitive Roles: Set to 9+
- Standard Roles: Keep at 8+
- High-Volume Hiring: Lower to 7+
- Specialized Roles: Add secondary criteria

Modifying the Form

To add additional fields to the form:
1. Click on "Application form" node
2. Click "Add Field"
3. Options include:
   - Text input (single line)
   - Textarea (multi-line)
   - Number
   - Dropdown
   - Date
   - Checkbox

Useful additional fields:
- Salary expectations
- Notice period
- Work authorization status
- Preferred start date
- Portfolio URL

Testing & Validation

Pre-Launch Testing Checklist

Test 1: Form Submission
- [ ] Form loads correctly
- [ ] All fields are visible
- [ ] File upload accepts PDFs
- [ ] Required validation works
- [ ] Form submits successfully

Test 2: PDF Extraction
- [ ] Text extracts from various PDF formats
- [ ] Formatting is preserved reasonably
- [ ] Multi-page PDFs work correctly
- [ ] Large files (>5MB) process successfully

Test 3: Profile Analysis
- [ ] AI correctly identifies names
- [ ] Contact info is extracted accurately
- [ ] Work experience is structured properly
- [ ] Skills are categorized correctly
- [ ] Education is parsed accurately

Test 4: Evaluation Quality

- [ ] Scores are reasonable (not all 10s or 0s)
- [ ] Strengths are specific and accurate
- [ ] Concerns are legitimate
- [ ] Recommendations make sense
- [ ] All fields are populated

Test 5: Filtering Logic

- [ ] Qualified candidates route correctly
- [ ] Score threshold works as expected
- [ ] Data formatting is clean
- [ ] Timestamps are accurate

Testing Commands

In n8n:

1. Click "Test Workflow" (top right)
2. Submit test data through form
3. Watch execution in real-time
4. Click on each node to see results
5. Verify JSON output at each stage

API Cost Monitoring:

Check your OpenAI usage:

1. Visit https://platform.openai.com/usage
2. Monitor costs during testing
3. Typical test: $0.05-0.15 total

Deployment

Going Live

Step 1: Final Checks

- [ ] All credentials are correct
- [ ] Error handling is in place
- [ ] Test with real resumes
- [ ] Verify scoring accuracy
- [ ] Check output formatting

Step 2: Activate Workflow

1. Toggle workflow "Active" (top right)
2. Click on Form Trigger node
3. Copy the Form URL
4. Share with candidates or embed on website

Step 3: Monitoring Setup

Set up alerts for:
- Workflow failures
- High error rates
- Unusual execution times
- API cost spikes

In n8n Cloud:
1. Go to Workflow Settings
2. Enable "Error Workflow"
3. Set up email notifications

For Self-Hosted:
1. Configure webhook for errors
2. Set up logging
3. Monitor server resources

Scaling Considerations

For <50 resumes/week:
- Default settings work fine
- n8n Cloud starter plan sufficient
- OpenAI costs: ~$1-5/week

For 50-200 resumes/week:
- Enable workflow queuing

- Monitor API rate limits
- Consider dedicated n8n instance
- OpenAI costs: ~$5-20/week

For 200+ resumes/week:
- Use queue mode
- Implement batch processing
- Dedicated infrastructure
- Consider GPT-4-mini for cost savings
- OpenAI costs: ~$20-100/week

Troubleshooting

Common Issues & Solutions

Issue 1: "Form not capturing file"

Symptoms: Binary data missing after form submission
Solution:
1. Check form field type is "File"
2. Verify accept types: ".pdf"
3. Ensure file size is under 16MB
4. Test with smaller PDF first

Issue 2: "PDF extraction returns empty"

Symptoms: Text field is empty or null
Solutions:
- If PDF is scanned: Use OCR pre-processing
- If PDF is protected: Remove password protection
- If PDF is corrupted: Re-generate the PDF
- If >100 pages: Split into smaller sections

Issue 3: "AI returns incomplete profiles"

Symptoms: JSON missing fields or has null values
Solutions:
1. Increase OpenAI token limits:
   - Max Tokens: 4000
   - Temperature: 0.3
2. Simplify JSON schema (remove optional fields)
3. Improve resume quality (ensure text is readable)
4. Check OpenAI API status

Issue 4: "Inconsistent scoring"

Symptoms: Similar candidates get very different scores
Solutions:
1. Set temperature to 0 for consistency
2. Refine evaluation prompt with specific criteria
3. Add examples to system message
4. Use GPT-4 instead of GPT-4-mini

Issue 5: "High API costs"

Symptoms: Unexpected OpenAI charges
Solutions:
1. Switch to GPT-4-mini: Saves 90% cost
2. Reduce token limits
3. Implement caching for repeated job descriptions
4. Optimize prompts to be more concise

Issue 6: "Workflow times out"

Symptoms: Execution fails after 2-5 minutes
Solutions:
1. Enable "Retry on fail" for AI nodes
2. Increase workflow timeout (Settings > Execution)
3. Process resumes in smaller batches
4. Upgrade n8n instance resources

Issue 7: "Merge node not working"

Symptoms: Data not combining correctly
Solutions:
1. Verify both inputs are receiving data
2. Check "Merge by Index" is selected
3. Ensure "Wait for all messages" is enabled
4. Test each branch independently

Issue 8: "Filter removing all candidates"

Symptoms: No candidates passing through
Solutions:
1. Check condition logic (>= vs >)
2. Verify field name matches exactly
3. Test with lower threshold temporarily
4. Review evaluation outputs manually

License & Usage

- Modify for your specific needs
- Deploy in production environments
- Share with your organization
- Integrate with other systems

Please attribute when sharing publicly

Resources

→ Youtube: YouTube Demo Episode

→ Newsletter: Tech Vibes with Shaphy Newsletter