Job Req Agent README

A reusable template-style README you can copy for future agents. It documents install locations, configuration, tools, data contracts, and runbooks based on the agent JSON you provided.

1) Overview

Name: Job Req Agent
Purpose: Transform a free-form job description into a structured, enriched, and market-validated posting, then publish to a job board.
Primary Users: Talent Acquisition, HR Ops, Recruiting Coordinators
Key Outcomes: Faster requisition turn-around, complete and compliant fields, competitive salary alignment, one-click publishing.

2) Install Location and Package Layout

After import/install, assets are placed under:

```
Bots/
  Demo Agents/
    Job Req Agent/
      Job Req Agent (agent definition)
      Tools/
        InputParser
        FieldEnrichment
        WebCompResearch
        PublishToJobBoard
        UserReview
        Dependencies/   (AI Skills used by the API Tasks are here)
```

3) Prerequisites

- Platform access to run AgentWorkflow and forms.
- Perplexity API keys for compensation research (WebCompResearch).
- Open AI GPT-4.1 Model Conenction (from Azure or OpenAI directly)
- Job board endpoint or internal Jobs Feed Board configured for publishing.

4) Configuration

Agent Variables

- **Input**
  - `Description` (string, required): Free-form job description from the hiring manager or HR.
- **Output**
  - `Result` (string): Final status message after publishing.

Model Connection

- **Vendor/Model:** OpenAI, `gpt-4.1` (named "GPT 4.1 - OpenAI")
- **Credential:** Secure storage of your API key to be used for the model connection

5) Typical End-to-End Flow

I say “typical” because this is a probabalistic solution - so it may use steps in a slightly different order dependent on the inputs received. Example - if no missing data is identified in the parse step, the agent will skip calling the UserReview form until the end.

1. **Parse** the free-form description with **InputParser** to extract structured fields and detect missing data.
2. **Review** any missing or ambiguous fields with the **UserReview** form.
3. **Enrich** content and standardize fields with **FieldEnrichment** (title, location, salary normalization; compliance polish).
4. **Research** market compensation with **WebCompResearch**; align salary ranges.
5. **Sync** salary mentions in the long description to the researched range.

6. **Final Review** in **UserReview**; confirm required fields are complete.
7. **Publish** to job board with **PublishToJobBoard**.
8. **Finish** with AgentWorkflowEvent.End (SUCCESS or ERROR).

6) Tools Catalog

> Each tool below includes path, purpose, inputs, outputs, and notes taken from your agent JSON.

6.1 InputParser

- **Path:** `Bots/Demo Agents/Job Req Agent/Tools/InputParser`
- **Command:** AgentWorkflow.Api
- **Purpose:** Convert free-form text into structured JSON, flag missing data for enrichment.
- **Inputs:**
  - `sDescription` (string, required): Freeform job description.
- **Outputs:**
  - `sJobData` (string, JSON): Working memory object with fields, extracted responsibilities/skills, missing fields, provenance.
- **Annotation:** Turn free-form text into structured data and highlight missing data.

Example output (abridged):

```
{
  "Title": "Developer Relations Manager",
  "Company": "",
  "Location": "Dayton, OH",
  "SalaryMin": "90000",
  "SalaryMax": "90000",
  "JobType": "Full-time",
  "Description": "Normalized role summary...",
  "DetectedResponsibilities": ["Run developer relations program", "..."],
  "DetectedSkills": ["Technical communication", "..."],
  "MissingApiFields": ["Company", "Department"],
  "Provenance": {"Title": "Inferred from description", "...":"..."}
}
```

6.2 FieldEnrichment

- **Path:** `Bots/Demo Agents/Job Req Agent/Tools/FieldEnrichment`
- **Command:** AgentWorkflow.Api
- **Purpose:** Enrich and standardize job data (title, location, salary, expanded description, compliance language).
- **Inputs:**
    - `sJobData` (string, JSON): Working memory from InputParser or prior step.
- **Outputs:**
    - `sJobData` (string, JSON): Updated working memory with cleaned fields and best-practice description.
- **Annotation:** Fix city/state, malformed content, and expand descriptions to meet compliance guidelines.

6.3 WebCompResearch

- **Path:** `Bots/Demo Agents/Job Req Agent/Tools/WebCompResearch`
- **Command:** AgentWorkflow.Api
- **Purpose:** Research market comps and compute a competitive salary range; adjust narrative mentions.
- **Inputs:**
    - `sJobData` (string, JSON): Current working memory.
- **Outputs:**
    - `sJobData` (string, JSON): Updated salary fields and synchronized description text.
- **Annotation:** Uses Perplexity web research to validate market competitiveness.
    - **NOTE: A Perplexity API Key is required for this step to work properly.**

6.4 UserReview (Form)

- **Path:** `Bots/Demo Agents/Job Req Agent/Tools/UserReview`
- **Command:** AgentWorkflow.Form (INTERACTIVE)
- **Purpose:** Human-in-the-loop review to fill missing fields and approve the posting.
- **Fields:**
    - `TextBox0` Company Name
    - `TextBox1` Job Title

    - `TextBox4` Job Type
    - `TextBox5` Location
    - `TextArea0` Role Description
    - `TextBox6` Salary Min
    - `TextBox7` Salary Max
    - `TextArea1` Provenance
- **Buttons:** Submit
- **Annotation:** Display and collect updates for all current job fields before publishing.

6.5 PublishToJobBoard

- **Path:** `Bots/Demo Agents/Job Req Agent/Tools/PublishToJobBoard`
- **Command:** AgentWorkflow.Api
- **Purpose:** Publish finalized job posting to the Job Feed job board.
    - By default, it publishes to: https://pathfinder.automationanywhere.com/challenges/training/jobfeed/index.html
- **Inputs:**
    - `sJobData` (string, JSON): Final approved posting.
- **Outputs:**
    - `sJobPostResults` (string, JSON): API result with post status and link if available.
- **Required Fields:** Title, Company Name, Location, Salary Min/Max, Job Type, Description.
- **Annotation:** Executes the publishing transaction and returns result.

6.7 End States (SYSTEM_TOOL)

- **SUCCESS:** AgentWorkflowEvent.End(status=SUCCESS)
- **ERROR:** AgentWorkflowEvent.End(status=ERROR)
- **CANCELLED:** AgentWorkflowEvent.End(status=CANCELLED)

7) Data Contracts

7.1 Input Contract

```
{
  "Description": "string (required) – free-form job description"
}
```

7.2 Working Memory (`sJobData`)

Key fields commonly present:

```
{
  "Title": "string",
  "Company": "string",
  "Location": "string",
  "SalaryMin": "string (numeric)",
  "SalaryMax": "string (numeric)",
  "JobType": "string",
  "Description": "string",
  "Department": "string",
  "DetectedResponsibilities": ["..."],
  "DetectedSkills": ["..."],
  "MissingApiFields": ["..."],
  "Provenance": {"<field>":"how derived"}
}
```

7.3 Publish Result

```
{
  "status": "SUCCESS|ERROR",
  "jobUrl": "string (if available)",
  "message": "string",
  "raw": { "apiResponse": "..." }
}
```

8) How To Run

1. Navigate to **Bots > Demo Agents > Job Req Agent**.
2. Provide the **Description** input (paste free-form JD).
3. Start the agent.
4. Complete **UserReview** form prompts when displayed.
5. Confirm final preview.
6. Agent publishes and returns **Result** with job board status or link.

9) Required and Validated Fields

- Job Title
- Company Name
- Location (City, State or country format standardized)
- Job Type (Full-time, Part-time, Contract, etc.)
- Salary Min, Salary Max (numeric strings; currency handled by job board context)
- Description (compliant and concise, includes EEO and accommodations text where applicable)

If any are missing, the agent will block publishing and route to **UserReview**.

10) Error Handling and Retries

- **Parsing Errors:** Return to UserReview to capture missing values.
- **Enrichment Failures:** Preserve original values; log details in provenance.
- **Comp Research Timeout:** Keep provided salary, flag for review, and notify in Result.
- **Publish Errors:** Surface API message in `sJobPostResults.message` , set End status ERROR.

11) Security and Governance

- Store secrets in credentials lockers; do not hard-code keys.
- The agent logs provenance explaining how each value was derived or validated.
- Use HITL (if enabled) for sensitive, ambiguous, or policy-gated scenarios.

- Keep compliance language templates in `Tools/Dependencies` if you need to standardize org-wide.

12) Performance and SLAs

- Typical end-to-end cycle time: minutes, driven by user review and web research latency.
- Parsing and enrichment steps are near-instant; market research may take longer based on external API responsiveness.

13) Customization Guide

Some areas where this could be improved with your testings and customization. Remember, as you add new tools/make significant tool modifications, you should also consider updating tool descriptions and potentially the agent definition/action plan.

- **Compliance Text:** Update description boilerplate in FieldEnrichment.
- **Field Enrichment:** Currently field enrichment uses simple text to enhance the job description. You could consider using RAG with a knowledge base of compliance text, company policies, and job descriptions to enhance the job description.
- **Salary Policy:** Add floor/ceil rules post WebCompResearch step.
- **Additional Fields:** Extend `sJobData` schema and add corresponding UserReview form fields.
- **Alternate Job Boards:** Swap or chain PublishToJobBoard with additional publisher tasks.

14) Testing

- Provide several sample descriptions with varied completeness:
  - Missing company name
  - City without state
  - Salary band implied vs explicit
  - Contract vs Full-time roles
- Verify:
  - MissingApiFields are discovered and resolved via UserReview.

    - Salary mention in long description is updated to researched range.
    - PublishToJobBoard returns a success link.

15) Known Limitations

- WebCompResearch depends on external data availability and may return coarse bands for niche roles or new geographies.
- Non-standard location formats may require manual correction during review.

16) Changelog (Template)

- **v1.0.0**
    - Initial release: parsing, enrichment, market comp, review, publish.

> For future updates, document added fields, policy changes, and any breaking schema changes to `sJobData`.