Automated Job Application Agent – Design Document

Version: 1.0
Audience: Cursor (code generation), maintainers
Status: Approved for implementation

---

1. Purpose & Scope

This document specifies the design for a compliant, human-in-the-loop system that automates job applications using a structured candidate profile and a public GitHub repository of internship postings. The system maximizes automation while respecting third-party ATS constraints (login, email verification, CAPTCHA).

In scope - Parsing a GitHub repository to extract job postings and application URLs - Detecting Applicant Tracking Systems (ATS) - Browser automation to open application pages - Automatic PDF resume upload - Automatic form autofill and submission - Human-in-the-loop pauses for login, email verification, and CAPTCHA - Logging, retries, and metrics

Out of scope (explicitly prohibited) - Email inbox scraping or OTP interception - CAPTCHA solving or bypassing - Automated account creation or identity spoofing - Headless evasion or ToS circumvention

---

2. High-Level Architecture

```
assets/                # Static files (resume PDF)
config/                # Configuration and mappings
src/
  ├─ ingest/            # GitHub repo parsing
  ├─ normalize/         # ATS detection & job normalization
  ├─ browser/           # Playwright orchestration
  ├─ autofill/          # Field mapping & form filling
  ├─ checkpoints/       # Human-in-the-loop pauses
  ├─ submit/            # Validation & submission
  └─ log/               # Logging & metrics
```

Primary technologies - Node.js + TypeScript - Playwright (non-headless by default) - Markdown parser for GitHub README

---

3. Inputs

3.1 Candidate Profile (JSON)

Single source of truth for all applications.

```
{
  "personal": {
    "first_name": "<string>",
    "last_name": "<string>",
    "email": "<string>",
    "phone": "<string>",
    "location": "<string>",
    "work_authorization": "<string>"
  },
  "education": [
    {
      "school": "<string>",
      "degree": "<string>",
      "field": "<string>",
      "graduation": "YYYY-MM"
    }
  ],
  "skills": {
    "languages": ["<string>"],
    "ml": ["<string>"],
    "tools": ["<string>"]
  },
  "links": {
    "github": "<url>",
    "linkedin": "<url>"
  }
}
```

3.2 Resume Asset

```
assets/
  resume.pdf
```

Optional metadata:

```
{
  "resume": {
    "file_path": "./assets/resume.pdf",
    "file_name": "Resume.pdf",
    "mime": "application/pdf"
```

```
  }
}
```

3.3 Job Source

Public GitHub repository containing internship postings in Markdown tables. The system fetches and parses README content to extract job data.

---

4. Job Ingestion & Normalization

4.1 GitHub Parsing

Responsibilities: - Fetch README.md - Parse Markdown tables - Extract: - Company - Role - Location - Application URL

Output schema

```
{
  "company": "<string>",
  "role": "<string>",
  "location": "<string>",
  "apply_url": "<url>",
  "source": "github"
}
```

4.2 ATS Detection

Classify application URLs using deterministic heuristics.

URL Pattern	ATS
greenhouse.io	Greenhouse
jobs.lever.co	Lever
myworkdayjobs.com	Workday
ashbyhq.com	Ashby
icims.com	iCIMS
otherwise	Custom

The detected ATS controls upload and autofill strategies.

---

5. Browser Automation Core

5.1 Session Management

- Persistent browser context
- Cookies retained across pauses
- No page reload during human checkpoints

5.2 Navigation Flow

```
Open apply_url
→ Login required? → Pause
→ Resume automation
→ Detect resume upload → Upload PDF
→ Autofill remaining fields
→ CAPTCHA? → Pause
→ Submit
```

---

6. Human-in-the-Loop Checkpoints

Automation must **pause and notify the user** when: - Login is required - Email verification is triggered - CAPTCHA is detected

Rules - Browser stays open - User completes the step manually - User resumes automation via CLI/UI signal

---

7. Resume PDF Upload

7.1 Detection

Use semantic and accessibility signals: - `<input type="file">` - Associated label text: Resume, CV, Upload, Attach - ARIA labels/descriptions

7.2 Upload Strategy

- Upload once per application
- Prefer resume upload over LinkedIn import
- Accept ATS auto-parsing defaults

7.3 Validation

Confirm upload success via: - Visible filename - “Uploaded / Attached” indicator - Absence of validation error

If upload cannot be detected: - Mark job as `manual_resume_upload_required` - Continue with other automation steps

---

8. Form Autofill Engine

8.1 Mapping Strategy

Map candidate profile fields to ATS fields using: - Label text - Placeholder text - Name/ID heuristics

8.2 Supported Field Types

- Text inputs
- Dropdowns / selects
- Radio buttons
- Checkboxes
- File uploads

8.3 Validation

- Detect missing required fields
- Attempt one corrective pass
- Avoid repetitive retries

---

9. Submission & Logging

9.1 Submission

- Final validation
- Click submit
- Confirm success (thank-you page or confirmation text)

9.2 Logging Schema

```
{
  "company": "<string>",
  "role": "<string>",
  "ats": "<string>",
  "status": "submitted | failed | partial",
  "issue": "<optional string>",
  "timestamp": "ISO-8601"
}
```

---

10. Error Handling & Safety

- Fail fast on repeated errors
- No aggressive retries
- No DOM mutation beyond form interaction
- No background automation during pauses

11. Success Criteria

- Resume PDF uploaded automatically in ≥90% of applications
- Human input required only for login, email verification, CAPTCHA
- Average application time ≤3 minutes
- Zero account lockouts or bans

---

12. Implementation Notes for Cursor

- Prefer semantic selectors over brittle CSS
- Implement ATS adapters as isolated modules
- Treat resume upload as mandatory when available
- Respect explicit out-of-scope constraints

Cursor should treat this document as authoritative.