**Project Plan: The IntelliApply Bot**

**Document Version:** 1.0 **Date:** June 26, 2025

**I. Project Overview**

**A. Mission Statement**

To create a robust, automated system that intelligently applies for jobs by combining a visual workflow orchestrator (n8n) with a powerful, AI-driven browser automation script (Python/Selenium/Gemini). The system will be capable of handling dynamic application forms with varying questions, layouts, and input types.

**B. Core Functionality**

* **Input:** A list of URLs to job application pages.
* **Process:** For each URL, the system will navigate to the page, analyze its content, use an AI model to determine the correct answers from a predefined knowledge base, fill out all required fields, upload necessary documents, and submit the application.
* **Output:** A log of successfully submitted applications, a record of any newly created login credentials, and notifications detailing the outcome of each attempt.

**II. Core Technologies**

* **Orchestration Platform:** n8n (for scheduling, triggers, logging, and notifications)
* **Browser Automation:** Python 3.10+ with the Selenium library.
* **AI Engine:** Google Gemini API (for natural language understanding and response generation).
* **Configuration & Secrets:** python-dotenv for managing API keys and credentials.
* **Dependencies:** google-generativeai, selenium, python-dotenv.

**III. System Architecture: The Hybrid Model**

The system is split into two primary components that work together: **The Orchestrator (n8n)** and **The Executor (Python/Selenium)**.

+---------------------------------------------------------------------------------+

| n8n Workflow (The Orchestrator) |

| |

| [1. Trigger] ----> [2. Fetch Jobs] ----> [3. Execute Python Script] ----> [5. Process Output] ----> [6. Notify] |

| (Schedule) (API/RSS) (Calls the Executor) | (Success/Fail?) (Email/Slack) |

| ^ | |

| | v |

| [4. Python Script Returns Result (JSON)] |

+---------------------------------------------------------------------------------+

|

| (Passes Job URL & API Key)

v

+---------------------------------------------------------------------------------+

| Python Application (The Executor) |

| |

| [Selenium Browser] <-----> [AI Engine (Gemini)] <-----> [Knowledge Base] |

| (The Hands & Eyes) (The Brain) (Your Personal Data) |

| |

+---------------------------------------------------------------------------------+

**IV. Modular Breakdown: The Python Application**

To ensure the code is manageable, maintainable, and scalable, the Python application will be broken into several distinct modules within a src directory.

**main.py (The Entry Point & Conductor)**

* **Purpose:** This is the main script that n8n will call. It acts as the internal orchestrator for the Python modules.
* **Responsibilities:**
  + Parse command-line arguments (e.g., --url, --headless).
  + Load configuration and secrets using the config module.
  + Initialize the BrowserHandler, AIEngine, and KnowledgeBase objects.
  + Execute the main application loop: find form elements, get answers from AI, fill the form.
  + Gracefully handle errors and close the browser.
  + Return a final JSON status message to n8n.

**src/config.py (Configuration Manager)**

* **Purpose:** To centralize all settings and secrets.
* **Responsibilities:**
  + Use dotenv to load environment variables from a .env file (e.g., GEMINI\_API\_KEY).
  + Define constants like file paths for the resume, cover letter, and knowledge base.
  + Hold browser settings (e.g., whether to run in headless mode).

**src/knowledge\_base.py (Your Personal Data Hub)**

* **Purpose:** To provide a simple interface for accessing your personal information.
* **Responsibilities:**
  + Load the profile.json file into memory.
  + Provide a function, e.g., get\_info(key), to retrieve a piece of data.
  + Provide a function to list all available data keys for the AI to reference.

**src/browser\_handler.py (The Selenium Specialist)**

* **Purpose:** To encapsulate all direct interactions with the Selenium WebDriver.
* **Responsibilities:**
  + Initialize and configure the Selenium WebDriver (e.g., setting user-agent).
  + Contain all navigation functions (Maps\_to\_url).
  + House the complex logic for scraping and parsing a form (get\_form\_questions), returning a list of question objects (e.g., {'label': 'First Name', 'type': 'text', 'element': WebElement}).
  + Contain specific action functions: fill\_text\_input(element, text), select\_dropdown\_option(element, option\_text), upload\_file(element, file\_path), click\_element(element).
  + Handle closing the browser (quit()).

**src/ai\_engine.py (The Gemini API Communicator)**

* **Purpose:** To handle all communication with the Gemini API.
* **Responsibilities:**
  + Initialize the Gemini client using the API key from config.
  + A primary function, find\_answer\_key(question\_label, knowledge\_base\_keys), that takes the text from the website and determines the corresponding key in your profile.json.
  + A secondary function, generate\_essay\_answer(prompt, context), for open-ended questions.

**src/utils.py (Utility Helpers)**

* **Purpose:** A place for miscellaneous helper functions used across the project.
* **Responsibilities:**
  + generate\_random\_password() for creating new logins.
  + save\_credentials(website, email, password) to append new logins to a CSV file.
  + Custom logging functions.

**V. The n8n Workflow Plan**

1. **Trigger Node:** Schedule trigger to run the workflow (e.g., every day at 8 AM).
2. **HTTP Request Node:** Fetches job listings from a source (e.g., an RSS feed from a job board or a custom API).
3. **Split in Batches / Loop Node:** Iterates through each job URL received from the previous step.
4. **Execute Command Node:** This is the key integration point. It will run the command: python main.py --url "{{$json.jobUrl}}". It will pass any necessary credentials from n8n's credential manager.
5. **IF Node:** Checks the output from the main.py script. The script should return a simple JSON like {"status": "success", "message": "Applied to Software Engineer at Google"}. The IF node checks if status == "success".
6. **Notification Nodes:**
   * **On Success:** A Send Email or Slack node sends a confirmation message.
   * **On Failure:** A different node sends an alert with the error message.
7. **Data Logging Node:** A Google Sheets or Airtable node that logs the application details (Date, Company, Role, Status) for tracking.

**VI. Development Roadmap: A Phased Approach**

This project will be built in manageable phases to ensure quality and prevent being overwhelmed.

* **Phase 1: Foundation & Setup (The Skeleton)**
  + Set up the Python project structure with all the empty files (main.py, modules, etc.).
  + Initialize git and create a .gitignore file (crucially ignoring .env and data/credentials.csv).
  + Write the config.py and knowledge\_base.py modules.
  + In browser\_handler.py, write the code to simply open a browser and navigate to a URL.
* **Phase 2: Simple, Static Automation**
  + Target a single, simple job application form.
  + Hardcode the logic in main.py to fill 2-3 specific text fields and submit the form. This validates the core Selenium functionality.
* **Phase 3: AI Integration & Dynamic Answers**
  + Build the ai\_engine.py module.
  + Implement the get\_form\_questions logic in browser\_handler.py.
  + Modify main.py to use the AI engine to map questions to answers from the knowledge base for text inputs.
* **Phase 4: Advanced Element Handling**
  + Add the functions to browser\_handler.py and the corresponding logic in main.py to handle dropdowns, radio buttons, checkboxes, and file uploads.
  + Implement the logic for creating and saving new logins.
* **Phase 5: n8n Orchestration & Finalization**
  + Build the full n8n workflow as described in Section V.
  + Thoroughly test the integration between n8n and the Python script.
  + Implement robust error handling and logging throughout the entire system.

**VII. Important Considerations**

* **Ethics & Terms of Service:** This tool should only be used on websites where automated applications are not explicitly forbidden. The goal is to assist, not to spam.
* **Security:** API keys and credentials must be stored securely in the .env file and n8n's credential manager, and never committed to version control.
* **Quality Control:** The AI's responses must be validated. The system should be designed to flag ambiguous questions for manual review rather than submitting a potentially incorrect answer.