# **Product Requirements Document: Personal Research Agent for Job and Networking Leads**

## **1. Introduction & Vision**

### **1.1. Project Overview: Personal Research Agent for Job and Networking Leads**

This document outlines the requirements for the "Personal Research Agent for Job and Networking Leads" (hereafter referred to as the "Research Agent" or "the tool"). The primary objective of this project is to develop a software tool that automates in-depth research into alumni from specified educational institutions (Questrom, University School) and product managers based in New York City. The Research Agent is envisioned as a personal assistant to streamline and enhance the process of generating qualified leads for job opportunities and professional networking.

### **1.2. Problem Statement: Addressing Inefficiencies in Manual Job/Networking Research**

The core problem this project aims to solve is the highly time-consuming, repetitive, and often overwhelming nature of manual research for job and networking leads, particularly through platforms like LinkedIn. The user has acknowledged engaging in "a lot of manual LinkedIn scouting and networking outreach," a process known for its inefficiencies.

This individual experience is reflective of broader frustrations among job seekers. The job application process is frequently described as "extremely time-consuming".1 Many individuals resort to cumbersome methods like maintaining Excel spreadsheets to keep track of applications and proprietary company information.1 A common sentiment is the experience of "wasting hours on LinkedIn and Indeed only to find out that the best way to apply for jobs is on individual company websites".1 The entire endeavor can feel like a "job itself," characterized by the laborious tasks of tracking applications, repeatedly retyping the same information, and often facing a lack of response or "ghosting" from potential employers.1 The prevalence of these pain points across a wide range of job seekers underscores that the manual process is not merely a personal inconvenience but a widely recognized systemic inefficiency. This shared experience strongly validates the potential benefits of an automation tool like the proposed Research Agent, positioning it as a solution to a significant and common challenge.

### **1.3. User Goal: Automate Research for Curated, Contextualized Job and Networking Leads**

The primary goal for the user in undertaking this project is to automate the laborious and repetitive aspects of their research. This automation is intended to yield a curated list of job and networking leads. Crucially, these leads should be enriched with contextual insights, such as the identification of mutual connections, an understanding of a company's product focus, and data on company size. This contextual information is vital for personalizing outreach and assessing the potential fit of an opportunity or contact. The strategic intent behind this tool is clear: "It’s not for your job—but it’s about landing your next one."

### **1.4. Product Vision: A Smart, Automated Assistant to Streamline the Job Search and Networking Process**

The vision for the Research Agent is that of an intelligent, automated assistant designed to significantly streamline the user's job search and networking activities. In its ideal end-state, the tool will proactively identify and present high-quality job and networking opportunities tailored to the user's specific criteria and background. By automating the most tedious aspects of the research process, the Research Agent aims to free up the user's time and mental energy, allowing for a more focused and strategic approach to career development. The ultimate aim is to transform the search for leads from a reactive, manual slog into a proactive, data-driven pursuit, providing not just raw data, but actionable intelligence.

## **2. Target User Profile**

### **2.1. Primary User Description**

The primary user of the Research Agent is an individual with a proactive approach to career development. This user likely possesses a background in or a strong aspiration towards product management, evidenced by their focus on PM roles and their familiarity with relevant technologies (e.g., Python, APIs, web scraping concepts). They are alumni of Questrom and University School and are specifically targeting product management opportunities and contacts within the New York City area. Their technical proficiency suggests an ability to understand and potentially contribute to the development or refinement of such a tool.

### **2.2. Key Needs & Pain Points (Elaborated)**

The user's requirements and the problems they face drive the core functionality of the Research Agent:

* **Need for Efficiency:** A primary need is to drastically reduce the time and effort spent on manual, repetitive tasks. These include searching LinkedIn for relevant profiles, cross-referencing these profiles with job boards, and identifying pertinent contacts within target companies or networks.
* **Need for Curation:** The user seeks to overcome information overload by receiving a filtered, manageable list of leads. These leads should align with specific, predefined criteria such as mid-level product management roles, shared alma mater, common interests, and location within NYC.
* **Need for Context:** To make informed decisions and personalize outreach, the user requires quick, actionable insights about each lead. This includes information like mutual connections, the company's primary product focus, and an indication of company size.
* **Pain Point: Targeting the Wrong People:** A common pitfall in manual networking is investing time in outreach to individuals who are not the most relevant or helpful contacts for one's specific goals.2 The Research Agent aims to mitigate this by pre-filtering leads based on criteria like alma mater, role, and location, thereby increasing the likelihood of connecting with suitable individuals.
* **Pain Point: Vague Asks/Outreach:** Effective networking often hinges on clear and specific requests.2 While the Research Agent will not compose outreach messages, the contextual information it provides (e.g., mutual connections, company's product focus, open roles at the contact's company) empowers the user to craft more targeted, relevant, and therefore more effective "asks." The ability of the tool to identify mutual connections and alma mater links directly supports overcoming these common networking mistakes. It helps shift networking from a broad, often inefficient activity to a more targeted, strategic one by surfacing individuals with pre-existing, albeit sometimes distant, connections, allowing for more personalized and less "vague" outreach.
* **Pain Point: Information Silos:** Manually tracking leads, notes, and application statuses across disparate platforms (LinkedIn, job boards, personal notes, spreadsheets) is inefficient and prone to error. The user's current method of using Excel lists for tracking applications highlights this issue.1 The Research Agent, particularly with its Notion export feature, aims to centralize relevant information, providing a more cohesive overview of networking and job search activities.

## **3. Core Features & Functionality**

### **3.1. Data Ingestion & Processing**

#### **3.1.1. LinkedIn Data Acquisition (Alumni, NYC PMs)**

The Research Agent will acquire data from public LinkedIn profiles based on specific targeting criteria:

* Alumni of Questrom.
* Alumni of University School.
* Product Managers currently working in New York City.

The key public data points to be extracted from LinkedIn search engine result pages (SERPs) or public profiles include:

* Full Name
* Public Profile URL
* Current Job Title (e.g., "Product Manager," "Senior Product Manager")
* Current Company Name
* Company Location (to confirm NYC for PMs, if available)
* School/University Name(s) (for alumni matching)
* Headline
* Location (City, State - for NYC PMs)
* Mutual Connections (if visible on public SERPs; this is often limited without being logged in or having a direct connection. The tool should attempt capture if feasible from SERPs).
* Brief summary/description (if available on SERP snippet).

It is important to recognize that while extensive data points are available on full LinkedIn profiles (such as skills, detailed work experience, endorsements) 3, scraping public SERPs may yield a more limited dataset, typically including name, title, company, URL, and a snippet of their summary.5 Reliably accessing deeper details like a comprehensive list of skills or full work history from SERPs alone is challenging and would likely necessitate navigating to individual profiles. This action significantly increases the complexity of the scraping operation and elevates the risk of detection by LinkedIn. Therefore, the initial version of the Research Agent should prioritize the robust and reliable extraction of core data available directly from SERPs.

**Technology Choices & Considerations for LinkedIn Scraping:**

* **Selenium:** This browser automation tool allows for mimicking human behavior, handling JavaScript-heavy pages, and navigating complex user interactions, which are characteristic of platforms like LinkedIn.6 If direct scraping of LinkedIn is pursued, Selenium is a common choice.
  + *Pros:* Capable of accessing dynamically loaded content and performing actions like a human user.
  + *Cons:* Generally slower and more resource-intensive than other methods. It can be more detectable by anti-scraping measures if not configured with care, as LinkedIn actively employs technical safeguards to prevent automated access.6
* **SerpApi (or similar proxy/scraper services e.g., Scrapingdog, BrightData):** These services manage the complexities of web scraping, such as proxy rotation, CAPTCHA solving, and maintaining browser fingerprints. This reduces the direct burden on the user's IP address and local infrastructure.5
  + *Pros:* Simplifies the process of bypassing anti-scraping mechanisms and can be more scalable. Scrapingdog, for instance, offers a LinkedIn Profile Scraper API that claims to operate without user cookies and is GDPR compliant for public data extraction.8
  + *Cons:* These services typically involve subscription costs. The user becomes reliant on a third-party service, and the specific data points returned are dictated by the capabilities of that service's API. For example, the Scrapingdog API mentioned does not extract job data from LinkedIn.8
* **Python requests with BeautifulSoup:** This combination is suitable for parsing HTML content from SERPs if the pages are relatively static or if a scraping service returns raw HTML.5
  + *Pros:* Lightweight and fast for processing static HTML content.
  + *Cons:* Generally ineffective for websites with dynamic content loaded by JavaScript or those requiring login or complex interactions, unless paired with sophisticated session management and anti-detection strategies.

Ethical and Legal Considerations for LinkedIn Scraping:

The act of scraping LinkedIn data operates in a legally and ethically complex environment.6

* **Public Data Focus:** Scraping activities should be strictly limited to publicly available data. LinkedIn's User Agreement explicitly prohibits scraping.7 However, some legal interpretations and case law, such as *hiQ Labs v. LinkedIn*, suggest that scraping data that is publicly accessible on the internet may not be illegal under statutes like the Computer Fraud and Abuse Act (CFAA).7 Despite this, violating LinkedIn's Terms of Service can still lead to consequences such as account restrictions or IP blocking.
* **Respect robots.txt:** It is a widely accepted ethical practice to respect the robots.txt file of a website, which provides directives for web crawlers.11
* **Rate Limiting and Human-like Behavior:** To avoid detection and overburdening LinkedIn's servers, it is crucial to limit the frequency of requests and implement delays to mimic human browsing patterns.6
* **Data Privacy:** Collection of personal data must be handled responsibly. Avoid collecting sensitive personal information, and ensure that any data collected is used ethically and in compliance with privacy regulations like GDPR where applicable.11 The fact that this project is intended for personal use might carry different implications compared to commercial scraping activities, but caution and adherence to ethical best practices remain paramount.

#### **3.1.2. Company Page Information (Contextual Insights)**

If a lead's current company is identified, the Research Agent could optionally attempt to fetch basic public information about that company. This data can be sourced from LinkedIn company pages or other public business directories. The purpose is to provide contextual insights such as company size (employee count), industry, and a brief overview of its product focus or mission (e.g., from an "About Us" snippet).4 This would typically involve a secondary scraping operation, contingent on the successful identification of the company.

#### **3.1.3. Job Board Integration (Lever, Greenhouse)**

A key feature is to cross-reference identified individuals or their companies with open product management roles in NYC, utilizing APIs from popular Applicant Tracking Systems (ATS) like Lever and Greenhouse.

* **Lever API Integration:** 13
  + **Endpoint for Listing Job Postings:** GET https://api.lever.co/v1/postings is available to retrieve job postings.13
  + **Filtering:** The Lever API supports server-side filtering of job postings by text (which can be used for job titles like "Product Manager") and by location (e.g., "New York City").13
  + **Key Data Fields:** Available data per job posting includes title, location, detailed descriptions, and URLs for viewing the job and applying.13 The company name might be inferred from the overall account context or linked data.
  + **Authentication:** Lever's API typically uses an API Key (via Basic Auth) for authentication, which is simpler for a personal tool, though OAuth is also an option.13
  + **Rate Limits:** The default rate limit is approximately 10 requests per second, with the possibility of bursting to 20 requests per second, though these are subject to change and server load.13
* **Greenhouse API Integration:** 15
  + **Endpoint for Listing Job Postings:** GET https://boards-api.greenhouse.io/v1/boards/{board\_token}/jobs is used to list job postings for a specific company's job board.16
  + **Filtering:** The Greenhouse Job Board API's primary endpoint for listing jobs does *not* offer direct server-side filtering by job title or location. To achieve this, the Research Agent would need to fetch all job postings for a given board\_token and then perform the filtering logic on the client-side (i.e., within the tool itself).16
  + **Key Data Fields:** Information available includes job title, location (as an object containing a name field), the full job description in HTML (if the ?content=true parameter is used), and the absolute URL to the job posting.16 The company name is implicitly known from the board\_token itself, as each token is specific to an organization.
  + **Authentication:** For GET requests to retrieve public job board data, authentication is typically not required.16
  + **Rate Limits:** Specific rate limits for the Greenhouse Job Board API are not explicitly detailed in the provided materials and would require further consultation of their comprehensive documentation.

Identifying Company-Specific API Tokens/Board IDs:

A significant operational consideration for the job board integration feature is the method of obtaining the company-specific board\_token (for Greenhouse) or an equivalent identifier for Lever.

* For **Greenhouse**, the board\_token is generally found by manually inspecting the URL of a company's career page that is powered by Greenhouse.16 Programmatic discovery of these tokens is not a feature directly supported by the Greenhouse API. While services like Apify offer scrapers that might extract Greenhouse jobs, this involves using their platform and may not directly yield the board token itself.17
* For **Lever**, the situation appears similar. The provided information does not detail a method for programmatically discovering a company's unique Lever API identifier from their career page.13 This presents a challenge: for the Research Agent to automatically check any newly discovered company for open roles, it will likely not know that company's specific job board token or identifier. This implies that the user might need to manually find and input these tokens for each company they wish to track, or the tool would require a highly sophisticated (and potentially fragile) secondary scraping or inference mechanism to attempt to discover these tokens from company career pages. This factor could limit the scalability of automatically checking a broad range of companies unless a curated list of company board tokens is maintained by the user or the tool.

### **3.2. Lead Curation & Contextualization**

#### **3.2.1. Filtering Logic**

The Research Agent will apply filtering logic to refine the raw data into a list of qualified leads:

* **Mid-level PM Roles:** This requires defining criteria for "mid-level." Based on common industry titles, this would include roles such as "Product Manager," "Senior Product Manager," and potentially "Product Owner" or "Lead Product Manager".18 Titles indicating entry-level positions (e.g., "Associate Product Manager," "Junior PM") or executive-level roles (e.g., "Director of Product Management," "VP of Product Management," "Chief Product Officer") should be excluded.18 The focus should be on generalist PM roles or those whose descriptions align with core product management responsibilities, such as managing the product lifecycle from concept to launch, defining strategy, and overseeing execution.19 While years of experience would be a useful filter, this data is often harder to reliably extract from initial SERP scrapes.
* **Alma Mater Match:** The tool will cross-reference the extracted school/university names from LinkedIn profiles against the user's specified alma maters: "Questrom" and "University School."
* **Shared Interests:** Automating the identification of shared interests based solely on public LinkedIn SERP data is challenging. This data is typically found deeper within a profile. Initially, "shared interests" might be a field the user manually tags in Notion or could be based on keywords found in the limited public summaries available on SERPs. This could be considered for future enhancement if deeper profile data access is implemented.
* **Location Match:** For product manager roles and contacts, the tool will ensure that the extracted or inferred location is New York City.

#### **3.2.2. Generating Contextual Insights**

To aid the user in prioritizing and personalizing outreach, the tool will generate contextual insights:

* **Mutual Connections:** If detectable from public SERP data, the tool will highlight any mutual connections between the user and the lead. However, the visibility of this information on public pages is often limited.
* **Product Focus (Company):** If company page data is successfully scraped (Section 3.1.2), keywords from the company's "About Us" section or product descriptions can be used to provide a brief overview of its product focus.
* **Company Size:** Employee count data, if available from a company page scrape, will be presented.
* **Open Roles at Company:** The tool will link to any relevant open PM roles found at the lead's company via the Lever or Greenhouse API integrations.

### **3.3. Output & Presentation**

The curated leads and their contextual information will be made available to the user through two primary methods:

#### **3.3.1. Notion Export**

The Research Agent will utilize the Notion API to export leads as new items into a user-specified Notion database.20 This allows for persistent storage and manual tracking by the user.

A proposed structure for the Notion database includes the following properties (columns):

* **Lead Name:** Text (Notion Title property)
* **LinkedIn Profile URL:** URL
* **Current Role:** Text
* **Company:** Text
* **Alma Mater Match:** Select or Multi-select (Options: Questrom, University School, None)
* **Location:** Text (e.g., NYC)
* **Source of Lead:** Select (Options: Alumni Search, NYC PM Search)
* **Mutual Connections:** Text or Number (if data is available)
* **Company Product Focus:** Text (brief summary from company scrape)
* **Company Size:** Text or Number (e.g., "100-500 employees")
* **Open PM Roles at Company:** URL (link to job posting on Lever/Greenhouse) or Text ("Yes -", "No")
* **Date Added:** Created Time (automatically set by Notion)
* **Status:** Select (User-defined for tracking, e.g., New, To Contact, Contacted, Informational Interview Scheduled, Follow-up Needed, Applied)
* **Notes:** Rich Text (for user's private annotations)

Authentication with the Notion API will be managed using an internal integration token, configured by the user.20 The tool will map extracted data to the appropriate Notion property types (e.g., text, URL, select, multi-select, date, rich text).20

The export to Notion, particularly with the inclusion of fields like 'Status' and 'Notes,' elevates the tool's utility beyond simple data aggregation. It effectively transforms the Notion database into a lightweight, personalized Customer Relationship Management (CRM) system. This enables the user to manage their networking outreach, track application progress, and maintain a history of interactions for each lead, thereby creating an interactive dashboard for their job search and networking pipeline.

#### **3.3.2. Local Dashboard (Streamlit or Flask)**

As an alternative or complement to Notion, leads can be displayed on a local dashboard.

* **Streamlit:** This Python library is well-suited for rapid development of data-centric web applications and dashboards.21
  + *Pros:* Offers a low-code approach, many pre-built UI components, and is excellent for quick visualization and interaction. Setting up pages, sidebars, loading data (e.g., from CSV or Pandas DataFrames), and using interactive widgets like select boxes for filtering are straightforward.21 Its ease of use for turning Python scripts into interactive dashboards is a key advantage.22
  + *Cons:* May be less flexible for highly customized UI/UX or very complex application logic compared to a full web framework like Flask.
* **Flask:** A more versatile micro web framework for Python.23
  + *Pros:* Offers greater flexibility and control, making it suitable for more complex applications or if fine-grained control over HTML, CSS, and JavaScript is required. Many pre-built Flask dashboard templates and extensions are available.23 Extensions like Flask-MonitoringDashboard demonstrate its capacity for building sophisticated tools.24
  + *Cons:* Has a steeper learning curve for dashboarding tasks compared to Streamlit and typically requires more boilerplate code to achieve similar UI functionality.

**Dashboard Features:** The local dashboard should display leads in a tabular format, allow for basic filtering and sorting based on key criteria (e.g., alma mater, role, company), show the contextual insights gathered for each lead, and potentially provide direct links to the lead's LinkedIn profile or the corresponding Notion page item.

**Decision Factor:** For a personal tool primarily focused on presenting curated data with some interactive filtering, Streamlit is likely the faster and more straightforward option to implement. If the user envisions more complex interactions, a highly custom user interface, or plans to add significant backend logic beyond data presentation, Flask would be the more appropriate choice.

### **Table 1: LinkedIn Profile Data Points (Public SERP Focus)**

| **Data Point** | **Typical HTML Element/Selector (Illustrative)** | **Extraction Method (e.g., BeautifulSoup, Selenium)** | **Reliability from Public SERP** | **Notes/Challenges** |
| --- | --- | --- | --- | --- |
| Full Name | span.entity-result\_\_title-text, span.actor-name | element.text | High | Usually complete. |
| Public Profile URL | a.app-aware-link (within result item) | element.get\_attribute('href') | High | Essential for further reference. |
| Current Job Title | div.entity-result\_\_primary-subtitle | element.text | Medium | Can be truncated; may include past roles if not specific. |
| Current Company Name | div.entity-result\_\_secondary-subtitle | element.text | Medium | Generally reliable but can be part of a longer string. |
| Location Snippet | div.entity-result\_\_secondary-subtitle (often combined with company) | element.text (requires parsing) | Medium | Often city-level, may not be precise. |
| Summary Snippet | p.entity-result\_\_summary (if present) | element.text | Low to Medium | Often truncated or not present on all SERP views. |
| Alma Mater Snippet | Text within result item if keyword matches | element.text (requires keyword search) | Low | Reliably extracting requires profile page visit. |
| Mutual Connections | Specific element if visible publicly | element.text | Very Low | Rarely visible on public SERPs without login/connection. |

## **4. User Stories**

The following user stories capture the key motivations and desired outcomes for the Research Agent:

* **4.1. Alumni Network Exploration:** "As a job seeker, I want to automatically find alumni from Questrom and University School on LinkedIn, so that I can identify potential networking contacts in my field or target companies."
* **4.2. NYC Product Manager Scouting:** "As a job seeker targeting PM roles in NYC, I want to automatically identify product managers working in NYC, so that I can understand the landscape and find individuals for informational interviews."
* **4.3. Identifying Mid-Level PMs:** "As a user, I want the tool to filter and highlight mid-level Product Managers, so I can focus my networking efforts on peers and those slightly ahead in their careers."
* **4.4. Cross-Referencing with Open Roles:** "As a user, when I see a promising contact, I want to quickly know if their company is hiring for PM roles in NYC, so I can tailor my outreach or apply directly."
* **4.5. Consolidating Leads in Notion:** "As a user, I want all identified leads and their contextual information to be exported to a Notion database, so I can track my interactions and manage my networking pipeline in one place."
* **4.6. Reviewing Leads on a Dashboard:** "As a user, I want to view my curated leads on a simple local dashboard, so I can quickly review new findings and decide on next steps without opening multiple browser tabs."

## **5. Technical Considerations & Proposed Architecture**

### **5.1. Detailed LinkedIn Scraping Strategy**

The approach to scraping LinkedIn data is critical due to the platform's sophisticated anti-scraping measures.

* **Choice of Tool:** A primary decision is whether to build a custom scraper or use a third-party service.
  + **Third-Party Services (e.g., SerpApi, Scrapingdog):** Recommended as a starting point, especially for SERP data.8 These services handle many complexities like proxy rotation and CAPTCHA solving, significantly reducing development overhead and the risk of being blocked.
  + **Direct Scraping (Selenium):** If direct scraping is preferred for greater control or to avoid service fees, Selenium is the appropriate tool.6 However, this path requires significant development effort and ongoing maintenance. There is a direct trade-off to consider: using third-party APIs like SerpApi or Scrapingdog incurs monetary costs but substantially reduces development time and the risk of encountering LinkedIn's blocking mechanisms.5 Conversely, building a custom Selenium scraper is cheaper in terms of direct service fees but demands a vastly greater investment in development time to build and, crucially, to maintain its reliability against LinkedIn's continuously evolving anti-scraping technologies.6 For a personal project, this balance between development effort/maintenance burden and monetary cost is a key decision point.
* **Mimicking Human Behavior (if using Selenium):** To reduce the likelihood of detection, the scraper must emulate human browsing patterns.6
  + Use realistic user-agent strings and rotate them periodically.
  + Implement randomized delays between requests and actions.
  + Navigate through pages naturally (e.g., by simulating clicks on "next" buttons rather than directly constructing paginated URLs if possible).
  + Employ proxy rotation, with residential proxies being generally more effective than datacenter proxies for evading detection.11 Tools like GoLogin can assist in managing multiple browser profiles with distinct fingerprints and proxies.6
* **Rate Limiting:** Whether using a third-party API or direct scraping, strict adherence to rate limits (either those imposed by the API provider or self-imposed conservative limits for direct scraping) is essential.
* **Handling CAPTCHAs and Blocks:** Third-party services are designed to handle these. For a custom Selenium scraper, encountering CAPTCHAs and blocks is a major challenge that may necessitate integration with CAPTCHA-solving services or the implementation of robust error handling, retry logic, and dynamic proxy rotation.
* **Session Management:** The goal is to scrape publicly available data. Attempting to access data that is less restricted when "logged in" (even to a public view) would require careful management of sessions and cookies. However, the primary approach should focus on data accessible to a guest user.
* **Parsing HTML:** Regardless of how the HTML content is obtained (via requests, Selenium, or a third-party API), BeautifulSoup is a robust Python library for parsing this content.5 Identifying stable CSS selectors or XPath expressions for the target data points is key. These selectors may need periodic updates if LinkedIn changes its page structure.
* **Targeting Public Search:** The scraper should utilize LinkedIn's public search functionality, accessible to guest users.9 This might involve constructing search URLs with appropriate query parameters or using specific guest-accessible API endpoints if available (e.g., https://www.linkedin.com/jobs-guest/jobs/api/seeMoreJobPostings/search? for jobs, as mentioned in 9).

### **5.2. API Integration Mechanics**

Interfacing with external APIs (Lever, Greenhouse, Notion) will be a core component.

* **Lever & Greenhouse Job Board APIs:** 13
  + The Python requests library is suitable for making HTTP calls to these APIs.
  + Implement proper authentication: API keys for Lever (Basic Auth) 13; no authentication is typically needed for Greenhouse GET requests for public job data.16
  + Parse JSON responses returned by the APIs.
  + Implement graceful handling of API rate limits, potentially using exponential backoff strategies for retries.
  + Include robust error handling to manage API downtime, invalid responses, or changes in API structure.
* **Notion API:** 20
  + Use an official Notion Python client library if one exists and is well-maintained, or use the requests library for direct HTTP calls.
  + Authentication will be via a Bearer token (the internal integration token provided by Notion).20
  + Construct JSON payloads according to the Notion API's schema for creating database pages (items). This involves mapping the extracted lead data to the correct Notion property types and structures.

### **Table 2: Job Board API Capabilities (Lever vs. Greenhouse)**

| **Capability/Feature** | **Lever API Details** | **Greenhouse API Details** | **Project Implementation Notes** |
| --- | --- | --- | --- |
| **List Job Postings** | GET https://api.lever.co/v1/postings 13 | GET https://boards-api.greenhouse.io/v1/boards/{board\_token}/jobs 16 | Both provide core listing functionality. |
| **Filter by Job Title** | Supported via text query parameter (server-side).13 | Not directly supported on list endpoint; requires client-side filtering of all results for a board.16 | Greenhouse implementation will require fetching more data initially and filtering locally. |
| **Filter by Location** | Supported via location query parameter (server-side).13 | Not directly supported on list endpoint; requires client-side filtering based on location.name in results.16 | Similar to title filtering, Greenhouse requires local filtering for location. |
| **Key Data Fields per Job** | Title, location, descriptions, apply URLs, company (inferred/linked).13 | Title, location object, HTML content (if ?content=true), absolute URL, company (from board).16 | Both provide necessary fields. Greenhouse content is HTML. |
| **Authentication** | API Key (Basic Auth) or OAuth.13 | None required for public job board GETs.16 | Lever requires API key management. |
| **Rate Limits** | ~10 req/sec (burst to 20).13 | Not explicitly stated in provided snippets; consult full documentation. | Implement retry logic with backoff for both. |
| **Board/Company ID Discovery** | Manual inspection of career page URL likely; not detailed as programmatic.13 | board\_token manually found from career page URL; not programmatic via API.16 | Significant challenge for scalability. User may need to provide these IDs, or a complex inference mechanism is needed. |

### **5.3. Data Storage (Beyond Direct Export)**

While the primary outputs are direct export to Notion or display on a transient dashboard, some intermediate local storage can be beneficial:

* **Optional Local Cache:** Scraped data (e.g., from LinkedIn, company pages) can be temporarily stored locally in formats like CSV files or an SQLite database. This can be useful for:
  + Debugging the parsing and processing logic.
  + Avoiding the need to re-scrape data if an export process fails or if the tool is run multiple times for testing.
  + Building resilience into the data pipeline. SQLite is noted as a database persistence option for some Flask dashboard templates, indicating its suitability for lightweight local storage.23

### **5.4. Frontend/Dashboard Implementation**

The choice of framework for the local dashboard depends on the desired complexity and development speed.

* **Streamlit:** 21
  + **Structure:** A typical Streamlit app consists of a single Python script (e.g., dashboard.py).
  + **Components:** Utilize Streamlit's built-in components such as st.title for page titles, st.sidebar for controls and filters, st.selectbox for filter selection, st.dataframe or st.table for displaying lead data, and st.markdown for formatted text and links.
  + **Data Handling:** Data can be loaded from local CSV/SQLite files or directly from the data processing module into Pandas DataFrames, which Streamlit handles efficiently.
* **Flask:** 23
  + **Structure:** For better organization, Flask projects often use Blueprints to modularize the application. Routes are defined to handle requests for different views or pages.
  + **Templating:** Flask uses the Jinja2 templating engine to render dynamic HTML pages.
  + **Data Handling:** Data processed in the Python backend is passed to the Jinja2 templates for rendering.
  + To accelerate UI development, consider using a pre-built Flask dashboard theme (e.g., AdminLTE, Datta Able, as listed in 23) which provide styled components and layouts.

### **5.5. Configuration Management**

To maintain security and flexibility, sensitive information and configurable parameters should not be hardcoded into the application's source code.

* Store API keys (Lever), Notion integration tokens, target school names, the target city (NYC), keywords for PM titles, and any other configurable settings in a separate configuration file (e.g., a .env file, a config.ini file, or a Python module like config.py).
* The application should load these configurations at runtime. This practice makes it easier to update settings without modifying the core codebase and protects sensitive credentials.

### **Table 3: Technology Stack Overview**

| **Component** | **Recommended Tool/Library** | **Key Rationale/Considerations** | **Alternative(s)** |
| --- | --- | --- | --- |
| **LinkedIn SERP Scraping** | SerpApi / Scrapingdog | Handles anti-scraping complexities, proxy management, CAPTCHAs; reduces development burden. | Python Selenium + BeautifulSoup (higher dev effort, riskier) |
| **Job Board API - Lever** | Python requests library | Standard for HTTP requests; direct interaction with official API.13 | - |
| **Job Board API - Greenhouse** | Python requests library | Standard for HTTP requests; direct interaction with official API.16 | - |
| **Data Export - Notion** | Notion API Client (if available) / Python requests | Official or well-supported client simplifies interaction; requests for direct control.20 | - |
| **Local Dashboard UI** | Python Streamlit | Rapid UI development for data-centric apps, many built-in components.21 | Python Flask (more flexible, higher dev effort for UI) 23 |
| **Core Logic/Orchestration** | Python | Primary language for all components, rich ecosystem of libraries. | - |
| **HTML Parsing** | Python BeautifulSoup4 | Robust and widely used for parsing HTML/XML content.9 | lxml (faster, but steeper learning curve for some) |
| **Local Data Cache (Optional)** | SQLite / CSV files | Lightweight local storage; SQLite for structured queries.23 | JSON files |

## **6. Data Privacy & Ethical Guidelines**

Adherence to data privacy principles and ethical guidelines is paramount for the development and operation of the Research Agent.

### **6.1. Navigating LinkedIn's Terms of Service**

It must be acknowledged that LinkedIn's User Agreement explicitly prohibits automated data collection methods like scraping.7 Violating these terms can lead to consequences such as IP address blocking or account suspension.

The legal landscape surrounding web scraping of public data is nuanced. The *hiQ Labs v. LinkedIn* case, for example, suggested that scraping publicly available data might not inherently violate laws such as the Computer Fraud and Abuse Act (CFAA).7 However, this legal precedent does not negate a platform's Terms of Service, nor does it eliminate all risks.

Given that the Research Agent is intended for personal, non-commercial use, its activity may be viewed differently from large-scale commercial scraping operations. Nevertheless, this does not grant immunity from LinkedIn's policies or potential technical countermeasures.

Recommendation: Prioritize scraping methods that are less likely to be detected (e.g., using well-managed third-party APIs designed for this purpose) or, if engaging in direct scraping, do so with extreme caution and be prepared for potential disruptions or the need to adapt strategies.

### **6.2. Best Practices for Scraping Public Data Responsibly**

The following best practices should be strictly adhered to 6:

* **Only Public Data:** Scraping activities must be strictly limited to data that is publicly visible on LinkedIn without requiring a login. Do not attempt to access private profiles, data behind authentication walls, or information that a user has explicitly set to private.
* **Respect robots.txt:** Before scraping any domain, including LinkedIn, the robots.txt file should be checked and its directives respected.11 While the legal enforceability of robots.txt is debated, adherence is a widely accepted ethical guideline for web crawlers.
* **Rate Limiting/Politeness:** Implement significant, randomized delays between requests to avoid overloading LinkedIn's servers.6 If possible, schedule scraping activities during off-peak hours. The goal is to be a "polite" scraper.
* **User-Agent Identification:** Use a legitimate, common browser user-agent string. Consider rotating user agents from a list of current, valid strings to further mimic diverse, human-like traffic.6 Avoid using default user agents from libraries like requests without modification, as these are easily identifiable.
* **Data Minimization:** Only collect the specific data fields that are essential for the tool's defined purpose.12 Avoid indiscriminate collection of all available data.
* **Transparency (Self):** As this is a personal tool, the user is inherently aware of the data collection. No external users are involved whose data is being processed unknowingly.

### **6.3. Ensuring Ethical Use of Collected Data**

The data collected by the Research Agent is intended solely for the user's personal networking and job searching assistance.

* The collected personal data must not be shared with third parties, sold, or repurposed for any means other than the user's direct job search and networking efforts.
* When the user decides to reach out to contacts identified by the tool, they should do so in a transparent, professional, and respectful manner. It is important to draw a clear boundary: while the tool automates the *discovery* of potential leads and relevant information, it should not automate the *outreach* itself. Effective networking relies on personalized, human engagement, including asking meaningful questions and diligently following up.2 The Research Agent's role is to empower the user with high-quality information to make these interactions more targeted and effective, not to replace genuine human connection with automated messaging, which could be perceived as impersonal or spammy.

## **7. Success Metrics (for the tool's utility to the user)**

The success of the Research Agent, from the user's perspective, can be measured by its ability to enhance their job search and networking efforts. Key metrics include:

* **7.1. Lead Generation Rate:** The number of relevant alumni and NYC-based product manager leads identified by the tool per operational run or on a weekly basis.
* **7.2. Time Saved:** An estimation (potentially subjective) of the reduction in manual research time achieved by using the tool compared to the user's previous manual methods.
* **7.3. Contextual Insight Quality:** A qualitative assessment by the user regarding the usefulness and accuracy of the provided contextual information (e.g., mutual connections, company product focus, open roles) in prioritizing leads and personalizing outreach.
* **7.4. Conversion Rate (Networking):** The number of successful networking interactions (e.g., informational interviews scheduled, meaningful conversations initiated) that originate from leads generated by the Research Agent.
* **7.5. Job Application Relevance:** The number of relevant job postings identified through the tool for which the user subsequently submits an application.

## **8. Future Considerations & Potential Enhancements**

Once the core functionality of the Research Agent is established and proven useful, several enhancements could be considered for future iterations:

* **8.1. Expanded Data Sources:** Integrate with other professional networking platforms (if APIs or ethical scraping methods are available), company review websites (e.g., Glassdoor for company culture insights), or tech news aggregators/blogs to gather timely information about target companies (e.g., funding rounds, new product launches).
* **8.2. Advanced Filtering Capabilities:** Introduce more granular filtering options for leads, such as filtering by specific skills (if deeper profile scraping is implemented and skills data becomes reliably available), years of experience, or company funding stage.
* **8.3. Automated Profile Summarization:** Employ basic Natural Language Processing (NLP) techniques to generate concise summaries of lengthy LinkedIn profiles or to automatically extract key skills, technologies, or themes mentioned in profile descriptions.
* **8.4. Enhanced Tracking and Workflow Integration:** Deepen the integration with Notion or potentially a calendar application to allow for tracking outreach attempts, setting reminders for follow-ups, and logging responses directly from the tool's dashboard or through a more seamless workflow.
* **8.5. Trend Analysis:** If the tool collects data consistently over time (and stores it appropriately), it could incorporate features to analyze trends, such as identifying which companies are most actively hiring product managers in NYC, or which skills are increasingly mentioned in job descriptions.
* **8.6. Support for More Job Boards:** Expand job board integrations beyond Lever and Greenhouse to include APIs from other popular platforms like Workable, Indeed, or LinkedIn Jobs (if a suitable API or scraping method is identified).
* **8.7. "Smart" Suggestions and Lead Scoring:** Implement a basic recommendation engine that, based on user activity (e.g., which leads are pursued, which job types are favored) or successful past interactions, suggests similar profiles, companies, or job opportunities. A simple lead scoring mechanism could also help prioritize leads.

## **9. Risks & Mitigation Strategies**

Several risks are associated with the development and operation of the Research Agent. Proactive mitigation strategies are essential.

* **9.1. LinkedIn Blocking/Detection:**
  + **Risk:** LinkedIn detects the scraping activity and blocks the user's IP address, restricts account access, or changes its website structure/anti-bot measures, rendering scraping tools or APIs ineffective.
  + **Mitigation:**
    - Prioritize the use of reputable third-party scraping APIs (e.g., SerpApi, Scrapingdog) that specialize in handling blocks and maintaining access.8
    - If direct scraping with Selenium is chosen, implement a comprehensive suite of anti-detection techniques: use rotating residential proxies, employ realistic and varied user agents, introduce random delays and human-like interaction patterns, and avoid predictable scraping patterns.6
    - Develop a fallback strategy or be prepared for periods of downtime if scraping methods break.
    - Regularly monitor the effectiveness of scraping scripts and be prepared to update them promptly in response to changes on LinkedIn's platform.
* **9.2. API Changes/Deprecation:**
  + **Risk:** The APIs for Lever, Greenhouse, or Notion undergo changes to their endpoints, data structures, authentication methods, or rate limits, or an API is deprecated, breaking the tool's integration.
  + **Mitigation:**
    - Write defensive code with robust error handling and logging around all API interactions.
    - If available, subscribe to API provider newsletters, developer blogs, or changelogs to stay informed about upcoming changes.
    - Implement periodic automated or manual testing of all API integrations to catch issues early.
    - Abstract API interaction logic into separate modules or classes to make updates and maintenance easier to manage.
* **9.3. Data Quality Issues:**
  + **Risk:** Scraped data is inaccurate, incomplete, outdated, or poorly parsed due to changes in website structure or flaws in the parsing logic.
  + **Mitigation:**
    - Implement robust and flexible parsing logic with thorough error checking and data validation.
    - For critical data points, if multiple sources were ever to be used (though less likely for this project's initial scope), consider cross-validation.
    - Ensure the Notion export allows for easy manual correction or annotation of data by the user if discrepancies are found.
    - Log parsing errors to help identify and fix issues with selectors or data formats.
* **9.4. Scope Creep for a Personal Project:**
  + **Risk:** The project's scope expands beyond the initial well-defined goals, leading to an overly complex tool that becomes difficult to complete or maintain, potentially causing user burnout.
  + **Mitigation:**
    - Strictly prioritize the implementation of core features that address the user's primary pain points first (Minimum Viable Product - MVP).
    - Treat all non-essential features and enhancements (Section 8) as items for future iterations, to be tackled only after the core functionality is stable and satisfactory.
    - Maintain a clear focus on the project's original objectives.
* **9.5. Legal/Ethical Missteps:**
  + **Risk:** The tool inadvertently violates LinkedIn's Terms of Service in a way that leads to more severe repercussions, or data is collected or handled in an unethical manner.
  + **Mitigation:**
    - Continuously review and strictly adhere to the ethical guidelines outlined in Section 6.
    - Prioritize scraping only publicly accessible data and always respect user privacy.
    - If there is any doubt about the permissibility or ethics of a particular data collection or usage practice, err on the side of caution and avoid it.
    - Keep abreast of discussions and legal precedents related to web scraping and data privacy.

## **10. Conclusion and Recommendations**

The Personal Research Agent for Job and Networking Leads project presents a valuable opportunity to address significant inefficiencies in the manual job search and networking process. By automating the discovery and initial contextualization of leads from LinkedIn and relevant job boards, the tool can provide the user with a curated and actionable dataset, saving considerable time and effort.

**Key Recommendations for Development:**

1. **Prioritize Robust LinkedIn Data Acquisition:** Given LinkedIn's anti-scraping measures, the initial focus should be on establishing a reliable method for obtaining public profile data. Utilizing a third-party API service (e.g., SerpApi, Scrapingdog) is strongly recommended for the initial version to mitigate blocking risks and reduce development complexity, even if it incurs a cost. If direct scraping is pursued, it should be approached with a deep understanding of the associated challenges and a commitment to implementing sophisticated anti-detection measures.
2. **Address Job Board API Identifier Discovery Early:** The challenge of obtaining company-specific board\_tokens (Greenhouse) or equivalent identifiers (Lever) needs a clear strategy. For an MVP, a manual input mechanism by the user for these tokens for target companies might be necessary. Future iterations could explore more automated (though complex) inference methods.
3. **Leverage Notion for Data Management and Workflow:** The Notion export feature should be designed not just as a data dump but as the creation of a functional mini-CRM. Structuring the Notion database with status tracking and note-taking capabilities will significantly enhance the tool's practical utility.
4. **Opt for Streamlit for Initial Dashboard:** For rapid development of a user-friendly local dashboard, Streamlit offers the most straightforward path. Its data-centric design aligns well with the project's core purpose of presenting curated leads.
5. **Maintain Strict Ethical Standards:** Adherence to ethical scraping practices, respect for data privacy, and responsible use of collected information must be non-negotiable throughout the project's lifecycle. This includes focusing only on publicly available data and using it solely for the intended personal job search and networking purposes.
6. **Iterative Development:** Begin with a core set of features (MVP) that address the most critical user needs: identifying alumni and NYC PMs, basic filtering, and exporting to Notion. Future enhancements can be added incrementally based on the tool's performance and the user's evolving requirements.

By focusing on these areas, the Personal Research Agent can become a powerful and efficient personal assistant, transforming a tedious manual process into a more strategic and data-driven approach to career advancement.

#### Works cited

1. What are your biggest pain points when trying to find a job? : r/GetEmployed - Reddit, accessed May 7, 2025, <https://www.reddit.com/r/GetEmployed/comments/1iyxepw/what_are_your_biggest_pain_points_when_trying_to/>
2. 5 Common Job Seeker Networking Mistakes (And How To Overcome Them), accessed May 7, 2025, <https://careerdirectors.com/job-seeker-networking-mistakes/>
3. How To Scrape Data From LinkedIn in 2025? Best Techniques - Evaboot, accessed May 7, 2025, <https://evaboot.com/blog/linkedin-data-scraping-2>
4. LinkedIn Datasets: Types, Applications and Providers [2025] - Research AIMultiple, accessed May 7, 2025, <https://research.aimultiple.com/linkedin-dataset/>
5. How to Scrape LinkedIn Data in 2025 - Bright Data, accessed May 7, 2025, <https://brightdata.com/blog/how-tos/linkedin-scraping-guide>
6. Web scraping Linkedin 2025: Pro Scraper's Guide + Code - GoLogin, accessed May 7, 2025, <https://gologin.com/blog/scraping-data-from-linkedin/>
7. Is LinkedIn Scraping Legal? Challenges & Laws Explained, accessed May 7, 2025, <https://www.bardeen.ai/answers/is-linkedin-scraping-legal>
8. LinkedIn Profile Scraper API & Company Page Data Extraction - Scrapingdog, accessed May 7, 2025, <https://www.scrapingdog.com/linkedin-scraper-api/>
9. How to Build LinkedIn Data Scraper with Python (Easy Guide) - ScraperAPI, accessed May 7, 2025, <https://www.scraperapi.com/web-scraping/linkedin/>
10. Scrape LinkedIn Profiles without login using Python - GeeksforGeeks, accessed May 7, 2025, <https://www.geeksforgeeks.org/scrape-linkedin-profiles-without-login-using-python/>
11. 8 Web Scraping Best Practices: A Guide to Responsible Data Extraction - DMNews, accessed May 7, 2025, <https://dmnews.com/8-web-scraping-best-practices-a-guide-to-responsible-data-extraction/>
12. Understanding the Legal Landscape of Web Scraping, accessed May 7, 2025, <https://web.instantapi.ai/blog/understanding-the-legal-landscape-of-web-scraping/>
13. Lever API Directory - Knit, accessed May 7, 2025, <https://www.getknit.dev/blog/lever-api-directory>
14. Lever API Integration - Knit Developer Docs, accessed May 7, 2025, <https://developers.getknit.dev/docs/lever-usecases>
15. greenhouse-api-docs/source/includes/job-board/\_applications.md at master - GitHub, accessed May 7, 2025, <https://github.com/grnhse/greenhouse-api-docs/blob/master/source/includes/job-board/_applications.md>
16. Introduction – Job Board API | Greenhouse, accessed May 7, 2025, <https://developers.greenhouse.io/job-board.html>
17. Greenhouse API in Python - Apify, accessed May 7, 2025, <https://apify.com/dlandberg/greenhouse/api/python>
18. Product Manager Job Titles in 2025 - Most Popular Titles by Career Level - Teal, accessed May 7, 2025, <https://www.tealhq.com/job-titles/product-manager>
19. Types of Product Management Roles: Finding Your Fit in a Diverse Field - Stanford Online, accessed May 7, 2025, <https://online.stanford.edu/types-product-management-roles-finding-your-fit-diverse-field>
20. Notion API integrations – Notion Help Center, accessed May 7, 2025, <https://www.notion.com/help/create-integrations-with-the-notion-api>
21. Building a dashboard in Python using Streamlit, accessed May 7, 2025, <https://blog.streamlit.io/crafting-a-dashboard-app-in-python-using-streamlit/>
22. Engineering Resources / Build a dashboard in Python with ClickHouse and Streamlit, accessed May 7, 2025, <https://clickhouse.com/engineering-resources/python-dashboard-streamlit>
23. Flask Dashboards - Admin Panels coded in Flask | App-Generator.dev - GitHub, accessed May 7, 2025, <https://github.com/app-generator/flask-dashboards>
24. flask-dashboard/Flask-MonitoringDashboard: Automatically monitor the evolving performance of Flask/Python web services. - GitHub, accessed May 7, 2025, <https://github.com/flask-dashboard/Flask-MonitoringDashboard>