Repository: gaurang-bhupendra-mhatre/wasserstoff/AIInternTask

🧠 Project Objective

The goal of this project is to build a browser automation tool capable of scraping public LinkedIn profiles based on a search query. The scraper intelligently navigates LinkedIn's search pages, extracts relevant profile data, caches visited content, and avoids redundant processing—all while imitating human browsing behavior to minimize detection.

⚙️ Tech Stack

Component	Technology
Programming	Python 3.10+
Automation	Playwright (Sync API)
Environment	dotenv
Anti-Detection	fake_useragent
Caching	JSON file-based system
Logging	Python logging module
Deployment	GitHub + Optional Vercel

🛠️ Features & Functionality

✅ Secure Login with Session Handling

- Automates the login process using Playwright.
- Supports session caching with cookies.
- Auto-reuses sessions within 12 hours to avoid redundant logins.

✅ Profile Scraping

- Extracts the following from search results:
    - Name

    - LinkedIn URL
    - Headline
    - Location
- Stops after reaching a user-defined profile limit.

✅ **Smart Caching**

- Avoids scraping duplicate profiles.
- Keeps track of visited search result pages.
- Saves data in a JSON-based cache (linkedin_cache.json).

✅ **Realistic Browsing Behavior**

- Introduces random delays between actions (2.5s to 6.5s).
- Rotates user agents to reduce bot detection.
- Detects and logs redirects or checkpoints.

🧩 **System Architecture**

pgsql

CopyEdit

```
User Input (Query, Email, Password)
        |
        ▼
 Start Playwright Browser
        |
        ▼
   Login to LinkedIn ──────────────▶ Load Saved Session (if valid)
        |
        ▼
  Go to Search URL → Scrape Profiles
```

```
     |                 ▲
     ▼                 |
  Save to Cache     Check if Already Visited
     |
     ▼
 Close Browser + Save Cookies
```

🧪 Run Instructions

🔧 Prerequisites

- Install dependencies:

bash

CopyEdit

```bash
pip install -r requirements.txt
playwright install
```

- Create a .env file with:

env

CopyEdit

```env
LINKEDIN_EMAIL=your-email
LINKEDIN_PASSWORD=your-password
```

▶️ Run Command

bash

CopyEdit

```bash
python scraper.py --email your-email --password your-password --query "Data Scientist" --headless --max 100
```

🗂️ Output Format

📁 Cache File: linkedin_cache.json

Stores all scraped data:

json

CopyEdit

```json
{
  "profiles": {
    "https://linkedin.com/in/example": {
      "name": "John Doe",
      "profile_url": "https://linkedin.com/in/example",
      "headline": "AI Researcher at XYZ",
      "location": "San Francisco, CA",
      "scraped_at": "2025-04-08T13:45:21"
    }
  },
  "visited_search_pages": [...],
  "cookies": [...],
  "last_session": "2025-04-08T13:40:00"
}
```

📝 Log File: linkedin_scraper.log

- Stores all runtime logs.
- Helps in debugging and tracking session behavior.

🔒 Anti-Bot & Detection Prevention

- Randomized user-agents.
- Time-based delays to mimic human browsing.
- Limits scraping to 200 profiles per session.

- Gracefully handles redirects and bot-check triggers.

🌐 Deployment & Hosting

- The code is public on GitHub.
- Can be hosted via **Vercel** or **Render** for dashboarding or visual front-end (optional extension).

🔄 Future Enhancements

- Add GUI with Streamlit or Flask for ease of use.
- Integrate with a database for persistent storage (e.g., SQLite, MongoDB).
- Add support for scraping deeper profile data after login.
- Export results as CSV/Excel.

📌 Conclusion

This project demonstrates browser automation, web scraping ethics, anti-detection strategies, and smart caching logic to efficiently gather data from LinkedIn's public search results. The modular and well-commented structure ensures maintainability and ease of extension in future iterations.