Job Collector System (Company → Roles → Spreadsheet)
A hands-free (read-only) job discovery system:
- Input: list of company names (or career URLs)
- Output: a spreadsheet (CSV/Excel/Google Sheet) with **Company + Role + Link + Location + Posted Date + Source + Match Score**
- You apply manually.

---

0) Goals & Non-Goals
Goals
- Automatically collect job postings from company career pages / ATS boards
- Filter by your target roles + location + seniority
- Export results to a spreadsheet with clickable links
- Run on-demand or on a schedule (daily/weekly)

Non-Goals
- NOT auto-applying to jobs
- NOT scraping LinkedIn (avoid ToS risk & instability)
- NOT bypassing CAPTCHAs or login walls

---

1) High-level Architecture
Pipeline
1. **Company Resolver**
   - Given `company_name`, find the likely job board URL(s) or accept a provided careers URL
2. **Source Detector**
   - Detect ATS type: `greenhouse`, `lever`, `ashby`, `smartrecruiters`, `workday`, `custom`
3. **Collector**
   - For ATS with public feeds: use **requests** + parsing (fast/stable)
   - For dynamic/custom pages: use **Playwright** fallback
4. **Normalizer**
   - Convert job postings from different sources into one standard schema
5. **Matcher / Filter**
   - Keyword match (title + location + optionally description)
   - Apply filters (remote/US/Houston; internship/entry; exclude senior)
6. **Exporter**
   - CSV/Excel OR Google Sheets
7. **Deduper**
   - Prevent duplicates and track new postings across runs

---

2) Data Model (Standard Job Schema)
Each job row should follow:

- `run_date` (YYYY-MM-DD)
- `company`
- `source` (greenhouse/lever/workday/custom)
- `board_url`
- `job_id` (best-effort unique)
- `title`
- `location`
- `remote` (true/false/unknown)
- `posted_date` (if available)
- `url` (job posting URL)
- `match_score` (0-100)
- `match_reasons` (e.g., "title: data engineer; location: remote")
- `notes` (optional, e.g., sponsorship keyword found)

---

3) Repo / Folder Structure
Recommended:

job-collector/
README.md
requirements.txt
.env.example
config/
companies.csv
keywords.yaml
src/
main.py
config_loader.py
resolver.py
detector.py
collectors/
base.py
greenhouse.py
lever.py
ashby.py
smartrecruiters.py
workday.py
custom_playwright.py
normalize.py
match.py
dedupe.py
export/
to_csv.py
to_excel.py
to_gsheets.py
data/
output.csv
runs/
2026-02-01.csv
cache/
resolved_boards.json
last_seen.json

---

4) Input Files
4.1 `config/companies.csv`
Minimum fields:
- `company`
- `careers_url` (optional; recommended if you already know it)

Example:
```csv
company,careers_url
Example Corp,https://boards.greenhouse.io/examplecorp
Another Inc,https://jobs.lever.co/anotherinc
```

```
WorkdayCo,https://workdayco.wd5.myworkdayjobs.com/en-US/WorkdayCoCareers
```

4.2 `config/keywords.yaml`

Example:

```
roles_include:
  - "data engineer"
  - "data analyst"
  - "analytics engineer"
  - "bi analyst"
  - "business analyst"
tech_keywords:
  - "sql"
  - "python"
  - "etl"
  - "power bi"
  - "snowflake"
  - "databricks"
  - "airflow"
  - "aws"

filters:
  location_mode: "usa_or_remote"   # options: remote_only | usa_or_remote | any
  preferred_cities:
    - "Houston"
  seniority_exclude:
    - "senior"
    - "staff"
    - "principal"
    - "lead"
    - "manager"
  internship_only: false
```

---

5) Tools & Libraries (Free)

Required

- Python 3.10+
- `requests`, `beautifulsoup4`, `lxml`
- `pandas`

For dynamic sites

- `playwright` (+ `playwright install`)

Optional (Google Sheets)

- `gspread`, `google-auth`

---

6) Collector Strategy (ATS-first)

Preferred order

1. Greenhouse (usually easiest)
2. Lever
3. Ashby
4. SmartRecruiters
5. Workday (often dynamic; sometimes needs Playwright)
6. Custom careers pages (Playwright + HTML parsing)

Collector interface (recommended)

Create a base class:

- `collect(board_url, company) -> list[JobRaw]`

Then normalize `JobRaw` into the standard schema.

---

7) Detection Logic (How to identify source)

Given `careers_url`:

- If domain contains `greenhouse.io` → greenhouse
- If domain contains `jobs.lever.co` → lever
- If domain contains `ashbyhq.com` → ashby
- If domain contains `smartrecruiters.com` → smartrecruiters
- If contains `myworkdayjobs.com` → workday
- Else → custom_playwright

If no `careers_url` provided:

- Resolver attempts to find likely board:
  - try `https://boards.greenhouse.io/{slug}`
  - try `https://jobs.lever.co/{slug}`
  - (or do a web search in a separate resolver module if you allow it)

---

8) Matching & Scoring (Simple but effective)

Title match

- +60 if title contains any `roles_include` phrase

Location

- +20 if remote detected and location_mode allows
- +10 if in USA
- +10 if city matches preferred (Houston)

Tech keywords (optional)

- up to +20 if keywords appear in description (if you fetch description)

Exclusions

- If title includes excluded seniority keywords → drop or strongly penalize

Output:

- `match_score` 0–100
- `match_reasons` string for transparency

---

9) Export Options

Option A: CSV (fastest)

- Write `data/output.csv`
- Also keep per-run snapshots `data/runs/YYYY-MM-DD.csv`

Option B: Excel

- `output.xlsx` with column formatting and clickable hyperlinks

Option C: Google Sheets (auto-updating)

- Use a service account JSON
- Write to a tab like `Jobs`
- Clear and rewrite each run OR append only "new jobs"

Recommendation:

- Start with CSV
- Add Google Sheets once everything works

---

10) Dedupe / "New Jobs" Tracking

Two ways:

Simple (per-run dedupe only)

- Deduplicate within the run on `(company, url)` or `(company, job_id)`

Persistent "new since last run"

- Maintain `data/cache/last_seen.json` mapping `url -> first_seen_date`
- Mark `is_new = true` if unseen in last_seen

Add columns:

- `is_new` (true/false)
- `first_seen_date`

---

11) Scheduling (Hands-free daily/weekly)

macOS / Linux: cron

Example daily at 9am:

```
0 9 * * * /usr/bin/python3 /path/job-collector/src/main.py
```

Windows: Task Scheduler

- Trigger daily
- Action: run python with script path

---

12) Step-by-step Build Plan (Milestones)

Milestone 1 — MVP (CSV output, ATS-only)

- Parse `companies.csv`
- Implement Greenhouse + Lever collectors
- Normalize to standard schema
- Filter/match on titles
- Export to CSV
- ✅ Works for a big chunk of companies

Milestone 2 — Add Workday + Playwright fallback

- Implement Workday collector (Playwright)
- Implement generic `custom_playwright` extractor for job cards/links
- Add retries + rate limiting

Milestone 3 — Better scoring + persistent "new jobs"

- Add description fetching where easy (ATS feeds)
- Add tech keyword scoring
- Add `last_seen.json` + `is_new`

Milestone 4 — Google Sheets output

- Add `to_gsheets.py`
- Write/append + keep formatting
- Add sheet tabs: `All Jobs`, `New Jobs`, `Companies`

---

13) Installation Commands (Local Mac)

```
python3 -m venv .venv
source .venv/bin/activate

pip install -r requirements.txt

# If using Playwright:
python -m playwright install
```

Example `requirements.txt`:

```
requests
beautifulsoup4
lxml
pandas
python-dotenv
playwright
```

Optional for Google Sheets:

```
gspread
google-auth
```

---

14) Run Commands

```
python src/main.py --input config/companies.csv --out data/output.csv
```

Useful flags (recommended to implement):

- `--min-score 60`
- `--remote-only`
- `--prefer-houston`
- `--export gsheets`

---

15) Practical Guardrails (To avoid blocks)

- Use modest concurrency (e.g., 2–4 companies at a time)
- Add random small delays between requests (0.5–2s)
- Cache resolved board URLs
- Prefer ATS feeds over browser automation
- If a site blocks, skip and log it (don’t brute force)

---

16) Deliverables Checklist

- `companies.csv` input
- ATS collectors (greenhouse, lever)
- Normalizer + job schema
- Matcher + filters
- CSV export
- Logging + error report
- Playwright fallback (workday/custom)
- Dedup + new jobs
- Google Sheets export (optional)
- Scheduler instructions

---

17) Notes for Antigravity Workflow

- Start by generating `src/collectors/greenhouse.py` + `lever.py`
- Build `normalize.py` + `export/to_csv.py`
- Test with 3–5 companies first
- Add Workday only after ATS collectors are stable

---

18) Next Info Needed From You (to tailor the code)

Paste these into `keywords.yaml` and `companies.csv`:

1. Company list (names + URLs if you have)
2. Role types you want (Data Engineer / Analyst / BI / etc.)
3. Location rules (Remote only / USA / Houston preferred)
4. Seniority preference (intern/entry/mid; exclude senior keywords)

Once you paste the company names, we'll implement collectors in the priority order that fits your list.

`::contentReference[oaicite:0]{index=0}`