**Crawler using OOP**

**Phase 1: Planning & Requirements**

* Define **goals**: Collect internship, research, and career development opportunities relevant to international students.
* Identify **essential data fields**:

**title, company, location, F-1/CPT/OPT eligible, link, type (internship, research, program).**

* Decide **scope**: full web is huge; focus on relevant domains first, with rules for discovering new URLs.
* Consider **constraints**: respect robots.txt, implement polite crawling, handle errors.

**Phase 2: Core OOP Design**

* **Base Crawler class**: fetch pages, store HTML, handle errors.
* **Frontier Manager class**: manage URLs to visit, track visited pages, implement prioritization.
* **Opportunity Data Model**: represent an opportunity with attributes and methods for validation/cleaning.
* **Site-specific parsers**: subclasses or plugins that know how to extract opportunities from specific page types.
* **Crawler Manager**: orchestrates crawling, parsing, and data storage.

**Phase 3: Page Fetching & Parsing**

* Fetch pages using requests or async libraries like aiohttp.
* Parse HTML with BeautifulSoup or lxml.
* Extract opportunity data **if page matches criteria**.
* Extract links for frontier expansion.

**Phase 4: Crawling Strategy**

* Implement **link-following**: BFS or DFS for visiting pages.
* Use **filters and heuristics**: keywords like “internship,” “research,” “international.”
* Prioritize **high-value domains** (universities, job boards, company career pages).
* Apply **politeness**: delays, rate limiting, respect robots.txt.

**Phase 5: Data Storage & Management**

* Store collected opportunities in a **central database** (SQLite, PostgreSQL, MongoDB) or CSV for smaller scale.
* Keep **metadata about visited pages** to resume crawling.
* Deduplicate entries to avoid multiple records for the same opportunity.

**Phase 6: Testing & Validation**

* Test on a small subset of sites first.
* Validate extracted data for completeness and accuracy.
* Handle exceptions: broken links, unexpected HTML, redirects.
* Ensure crawler doesn’t enter infinite loops or crawl irrelevant content.

**Phase 7: Scaling & Enhancements**

* Introduce **asynchronous or multithreaded crawling** for speed.
* Implement **priority queues** for URLs.
* Use **NLP or heuristics** to detect opportunity content automatically.
* Extend crawler dynamically with **new site parsers**.
* Optionally deploy in **cloud or distributed setup** for large-scale crawling.