# Impact Afghanistan Data Pipeline for Architecture, Sources, and Technical Lessons from Summer 2026 Development

## 1. Introduction

Impact Afghanistan is an ongoing project at Brown University built to keep Afghanistan visible in policy and research discourse at a moment even when international attention has largely moved elsewhere. Its central deliverable is an online platform that curates publications on and holds a database for publications about Afghanistan published after 2021, drawing on both institutional sources and civil society organizations working more directly with Afghan communities. The project also aims to identify where reporting on Afghanistan is thin or inaccessible, and it operates in partnership with Women for Afghan Women (WAW), a grassroots organization providing direct services to Afghan women and girls. It is explicitly aligned with the UN Summit of the Future, including the Global Digital Compact's commitments to more equitable digital information access.

My contribution this summer was building a pipeline that was python-based scrapers for UN Security Council, UN General Assembly, and UN Women sources and extracting the data into the corrected format to be able to be preserved on the website. This report covers the pipeline's architecture, the specific scraping and enrichment methodology used for each source, the technical failures encountered along the way — bot-protection, malformed links, missing metadata, a duplicate-import bug that silently doubled roughly two-thirds of the production database — and what each failure revealed about the constraints of automating this kind of archival work. It closes with recommendations aimed at whoever extends this pipeline next, since the project's stated goal of identifying reporting gaps is itself an ongoing task.

**2. System Architecture**

The pipeline separates into three stages: data acquisition, enrichment, and storage where each is handled by a different part of the codebase.

1) Acquisition. Scraping is done entirely in Python, using requests and BeautifulSoup for static pages and Selenium for JavaScript-rendered or bot-protected ones. Each source has its own script, since each site's structure (DOM), filtering, and access restrictions differ enough that a single generalized scraper wasn't realistic. After accessing each publication the output is written to CSV for one file per source  with the schema of source name, author, link, date of publication, country of publication, associated organizations, type of publication, an open-access flag, topics and summaries.
2) Enrichment. If the publications already don't have a given topics and/or summaries based on the already given metadata of the file, a separate pass adds a summary and a list of topics to each row for the missing variable, generated by calling the OpenAI API against either the full text of the source document (extracted with pdfplumber for PDFs, or from the raw HTML body for web articles) or, where full text was not reliably retrievable, a shorter description already provided by the source site. This step is decoupled from scraping where it operates directly on an existing CSV, checks whether a row already has real content in its summary and topics fields, and skips it if so.
3) Storage and serving. CSVs are committed to the project's Git repository, then imported into a MySQL database (afghanistan_db, publications table) via a Node.js script, import-csv.js. The application is a standard three-container Docker

Compose setup with MySQL, a Node/Express backend, and a React frontend with the backend exposing a single /data endpoint that returns the full publications table as JSON for the frontend to render and filter.

The architectural detail most worth flagging is the gap between the second and third where merging a pull request into the repository's main branch does not update the live site. The Git repository and the production database are two separate systems, connected only by someone manually running import-csv.js against production after a merge.

## 3. Data Sources and Methodology

Multiple sources were scraped this summer, each requiring a distinct approach. What follows documents each one's structure and the reasoning behind the technical choices made, guided throughout by the ethical and methodological considerations that apply to any research relying on scraped, publicly available data (Brown et al.).

### 3.1 UN Security Council and General Assembly

The Security Council scraper (UNDL_scrapper.py) covers five document types published on main.un.org/securitycouncil: Reports of the Secretary-General, Resolutions, Presidential Statements, Press Statements, and Notes by the President, each on its own yearly-URL listing page from 2021 onward. A sixth source, Meeting Records, is hosted separately on research.un.org and requires its own parsing logic, since its table carries seven columns rather than three that a single row may or may not populate.

All of these pages render their tables via JavaScript, so static requests return incomplete or empty content; Selenium was used throughout, loading each page in a headless Chrome instance before parsing with BeautifulSoup. Symbol-based links (e.g., S/2021/759) resolve to PDF text through the UN's document-access API, documents.un.org/api/symbol/access, which was used for full-text extraction ahead of summarization.

The General Assembly scraper follows a parallel but separately-built path against un.org/en/ga/{session}/resolutions.shtml, indexed by session number rather than by year (each GA session spans roughly September to September). Unlike the Security Council pages, this table loads statically but its content is denser where each row packs the draft resolution symbol, adoption date, vote outcome, and meeting record symbol into a single unstructured cell, requiring regular-expression extraction rather than clean column parsing.

### 3.2 UN Women Asia-Pacific

The UN Women's country page for Afghanistan links out to four sub-listings — stories, statements, summaries, and publications — each independently paginated. Two technical obstacles distinguished this source from the UN Security Council and General Assembly work. First, its listing pages are protected by a Varnish-based bot-detection layer that returns a 403 error on any direct, script-issued navigation to a filtered or paginated URL, even from a legitimate browser session; the same page loads without issue when a user clicks into it from a referring page. The scraper therefore navigates by simulating that click (locating and calling .click() on the actual DOM element) rather than requesting the target URL directly, and returns to the listing page via driver.back() after visiting each article rather than reconstructing the URL. Second, authorship on this source is inconsistent: some articles carry a named byline (identified

by locating a Author label and reading the text that follows it), while others, primarily institutional press releases, carry none, in which case the author field falls back to "UN Women." Where a fallback convention was needed, this project treated it as a bias-and-representativeness question, not merely a formatting one, following existing guidance on documenting how missing or inconsistent attribution can distort who appears to be "speaking" in a humanitarian dataset (Centre for Humanitarian Data).

### 3.3 Security Council Report

Security Council Report (securitycouncilreport.org) is an independent nonprofit that publishes its own commentary and analysis on Security Council activity. Its dedicated Afghanistan documents index groups entries into twelve categories (resolutions, presidential statements, Secretary-General's reports, sanctions committee documents, and others), each reachable through a "View All" link that filters the same underlying table by document type. This source was included specifically because it cross-references and, for some entries, corrects the Security Council data scraped directly from main.un.org.

## 4. Technical Challenges and Resolutions

### 4.0 The UN Digital Library blockade

Before any source specific scraper was built, the UN Digital Library (digitallibrary.un.org) was the obvious starting point. It indexes UN documents from every body, Security Council, General Assembly, and others, with structured metadata already filled in: title, author, date, and subject fields, consistent across record types. A single working scraper against

this one source would have covered most of the ground that four separate, source specific scrapers ended up covering instead.

Requests to the library were blocked at the same level described in the sections that follow, and no working method was found to retrieve its metadata programmatically. This was not a minor setback. It was the decision point that shaped the rest of the pipeline: without a central, structured source, each UN body's documents had to be collected from its own separate website, each with its own layout, its own quirks, and its own failure modes. The four source specific scrapers described in Section 3 exist because this one did not work. What follows in this section, bot protection on individual sites, malformed links, an import bug, are downstream problems that would have looked very different, and in most cases would not have existed at all, if the Digital Library had been usable from the start.

### 4.1 Bot-protection on institutional and NGO websites

Two of the sources actively block automated access, and neither block was uniform across the site. Countermeasures of this kind — rate limiting, IP blocking, CAPTCHA systems — are a documented, expected feature of the modern scraping landscape rather than an unusual obstacle specific to these two sites (Procedia Computer Science 259).

press.un.org, which hosts Security Council Press Statements, returns a 403 error to any request lacking browser-typical signals, regardless of whether the request comes from headless or non-headless Selenium, or from requests with a standard User-Agent header. This is consistent with a deliberate, well-configured anti-scraping system rather than an incidental JavaScript-rendering requirement, and no further evasion was attempted. The twenty-four affected rows are recorded with their real titles, dates, and links, but without generated summaries, and this limitation is documented rather than concealed. This decision follows the

broader convention that a site's technical countermeasures against automated access — like its robots.txt directives — represent a social norm scrapers are expected to respect even where no law strictly compels it (Gold and Latonero).

UN Women's listing pages presented a narrower version of the same problem: direct navigation to a filtered or paginated URL returned a 403 (again, a Varnish-layer response), but the identical page loaded successfully when reached by simulating a click on the actual link element from its referring page. This suggests the block was keyed to some property of direct, unreferred navigation rather than to automation generally, since a Selenium-driven click is no less automated than a Selenium-driven .get() call.

### 4.2 Malformed and placeholder links

A second class of problem was self-inflicted rather than imposed by the source site. The Security Council meeting-record scraper extracts outcome-document symbols from table cells that sometimes contain more than the symbol itself where because of the required format it was a resolution symbol followed by its vote tally, e.g., S/RES/2596 (2021) 15-0-0 and an early version of the extraction logic captured the entire cell's text rather than isolating the symbol from the <a> tag inside it. The result was a small number of unusable links carrying vote counts appended to what should have been a clean document identifier. This was corrected by extracting the anchor tag's text specifically rather than the cell's full text, and was tracked, alongside a related placeholder-link issue affecting a different scraper.

### 4.3 The duplicate-import incident

Because import-csv.js performed inserts without checking for existing records, re-importing a CSV (to correct data or add newly generated summaries) created a second copy of every row it contained rather than updating the first. To fix a database constraint paired with an upsert-based rewrite of import-csv.js was made at the infrastructure level rather than by asking every contributor to change their workflow.

**5. Data Quality and Enrichment**

Summarization and topic-tagging were treated as a genuinely separate stage from scraping, both because it depends on an external, metered service (the OpenAI API) and because that dependency changes what "safe to re-run" means. Every enrichment script checks a row's existing summary and topics fields before processing it and skips the row if both are already populated with real content. Without that check, re-running enrichment against a CSV that already had ninety percent of its rows correctly filled would reprocess all of it, at full API cost, to fix the remaining ten percent.

Topic tagging draws from a fixed taxonomy of ten main themes (Governance, Security and Conflict, Human Rights, Education, and others), each with several sub-themes, given to the model as part of the prompt alongside the source text; the model selects one to two main themes and one to two sub-themes per document rather than freely generating labels. This was a deliberate constraint: an open-ended topic-generation prompt produces vocabulary that drifts across documents processed at different times, which would make the resulting topics field far less useful for aggregation or filtering on the live site.

## 6. Recommendations for Future Contributors

Automate the deploy step. The gap between merging and deploying (Sections 2 and 4.5) is the single highest-leverage fix available to this project. A lightweight CI step that watches for changes to tracked CSV files and runs import-csv.js automatically on merge would eliminate an entire category of failure without requiring any change in how contributors work day to day. Until that exists, any contributor merging a CSV that touches already-live data should treat flagging that merge as part of finishing the task, not an optional courtesy.

Decide deliberately about bot-protected sources rather than revisiting them piecemeal. Two sources this summer (Section 4.1) turned out to be protected against automated access in ways that resisted every practical technique tried. That is a legitimate stopping point, not a failure to iterate further, but it should be recorded as a project-level decision rather than left implicit in a single contributor's code comments, where otherwise the same investigation is liable to be repeated by whoever next works on Security Council Press Statements or a similarly protected source. This matters more, not less, in a humanitarian context, where data governance is often self-regulated rather than externally audited — meaning that documenting a decision like this one is frequently the only mechanism ensuring it is actually followed (Lidén).

## 7. Conclusion

The work documented in this report is infrastructure and pipelines. The Impact Afghanistan project's stated goal is to identify where reporting on Afghanistan is thin or inaccessible; this summer's contribution was building the machinery capable of surfacing that gap systematically, across sources ranging from a formally structured UN resolution to a compilation of first-person testimony with no metadata at all. The most telling result may be less

any single dataset than the asymmetry between them: institutional sources, however bureaucratically dense, are largely built to be machine-read, while the grassroots testimony closest to lived experience in Afghanistan is the hardest to process automatically and the easiest to lose to a pipeline that isn't built to accommodate it. This asymmetry is not unique to this project; related work on dataset construction more broadly has found that materials gathered without deliberate intervention tend to systematically underrepresent exactly the populations and voices least likely to already have institutional infrastructure behind them (Jo and Gebru).

The technical failures described here (bot-protection, malformed links, unreliable dates, import bug) were each ordinary software problems with ordinary fixes. Taken together, they point to a more general condition of this kind of work: a small, resource-constrained team maintaining a growing, heterogeneous archive under real time pressure will accumulate this class of failure faster than it can be caught by any one contributor working alone. The recommendations in Section 6 are offered in that spirit — not as a final state for the pipeline, but as the next layer of infrastructure needed to keep pace with a project whose stated ambition, appropriately, exceeds what any single summer of work could complete.

**Works Cited**

Brown, Megan A., Andrew Gruen, Gabe Maldoff, Solomon Messing, Zeve Sanderson, and Michael Zimmer. "Web Scraping for Research: Legal, Ethical, Institutional, and Scientific Considerations." *arXiv*, 19 Dec. 2024, https://doi.org/10.48550/arXiv.2410.23432.

Centre for Humanitarian Data. *Guidance Note: Humanitarian Data Ethics*. OCHA, 2020, https://centre.humdata.org/wp-content/uploads/2020/02/guidance_note_ethics.pdf.

Gold, Zack, and Mark Latonero. "Tutorial: Legality and Ethics of Web Scraping." *Murray State University Faculty and Researcher Publications*, https://digitalcommons.murraystate.edu/cgi/viewcontent.cgi?article=1071&context=faculty.

Jo, Eun Seo, and Timnit Gebru. "Lessons from Archives: Strategies for Collecting Sociocultural Data in Machine Learning." *arXiv*, 13 Dec. 2019, https://arxiv.org/pdf/1912.10389.

Lidén, Kristoffer. "Ethics and the Governance of Digital Data in Humanitarian Action." *Disasters*, vol. 50, no. 1, 2026, e70018, https://doi.org/10.1111/disa.70018.

"Web Scraping: Legal and Ethical Considerations in General and Local Context — A Review." *Procedia Computer Science*, vol. 259, 2025, https://www.sciencedirect.com/science/article/pii/S1877050925012128.