**DELAWARE SCRAPING PROJECT**

**Goal: -**

To extract comprehensive data from all bid categories and their individual details pages.

**Objective:** -

The main aim is to use any library for extracting data from the Website which includes the following details.

* **From Main Table (for each bid in Open, Recently Closed, Awarded):**
  + Contract Number
  + Contract Title
  + Open Date
  + Deadline Date
  + Agency Code
  + UNSPSC
  + Current Bid Status (Open, Recently Closed, Awarded)
* **From Bid Detail Page (for each bid):**
  + Full Bid Title/Header
  + Solicitation Ad Date
  + Deadline for Bid Responses
  + Contact Name (if available)
  + Contact Email (if available)
  + List of all Supporting Bid Document URLs
  + Any other text information is present on the detail page.

**Approach: -**

* Launch the Delaware Bids Portal in a headless Chrome browser via Selenium and wait for the content to load.
* Navigate through all three categories (Open, Recently Closed, Not Awarded) by clicking their respective tabs.
* Paginate through all available pages in each category using the "Next" button.
* For each bid row: Extract high-level details: Bid ID, Contract Number, Title, Open, Date, Deadline, Agency, UNSPSC.
* Click on the bid **Title** to open its modal detail popup.
* Extract modal-specific information: Solicitation Ad Date, Deadline for Bid Responses, Contact Email, Important Message (if present), Supporting Bid Documents: Stored as a dictionary: {document\_name: document\_url}
* Close the modal popup using Escape key to return to the table view.
* Save the extracted data to an Excel file row-by-row using Pandas + OpenPyXL.
* Loads already-saved Bid IDs from the Excel file to skip reprocessing.
* Handles Ctrl+C or OS interrupts to ensure the browser quits cleanly.
* Uses Chrome in headless mode (--headless) for efficiency and automation on servers.
* Logs scraping events and errors to both **console** and a scraping\_log.txt file.

**Tools Used: -**

* Selenium

a. Options

b. Services

c. chrome Driver

d. by

e. Action Chains

f. EC

g. Web Driver

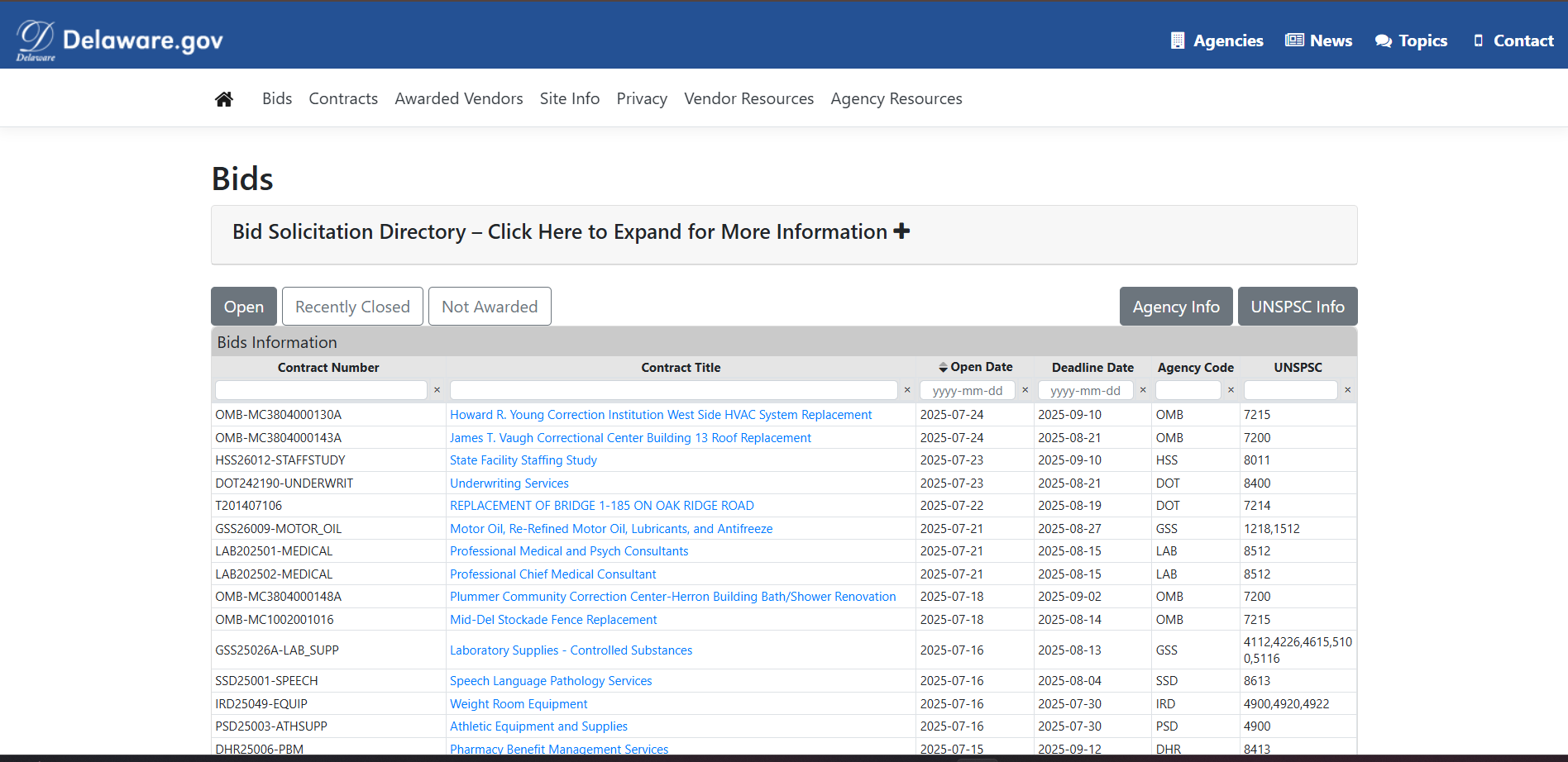
* Openpyxl
* Os
* Log
* Web Driver
* Time
* Pandas

**Problem faced: -**

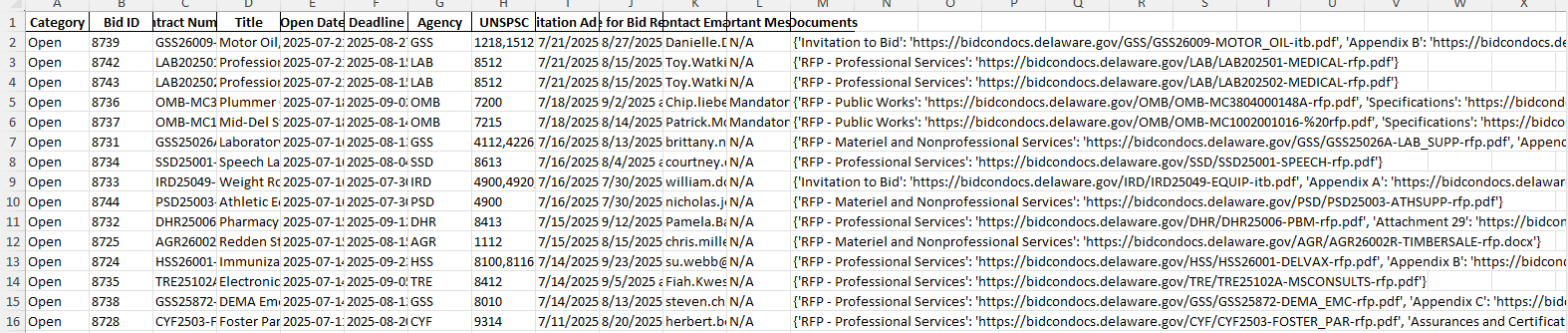
* Dynamic content loads slowly: Used WebDriverWait with expected\_conditions to wait for tables and modals to fully load.
* Used XPath with preceding-sibling::label to extract modal data based on visible labels.
* Used ActionChains to send Escape (ESC) key for consistent modal closing.
* Detected “Next” button by ID, checked if it’s disabled, and clicked using JavaScript to reliably move to the next page.
* Used openpyxl + ExcelWriter in append mode with if\_sheet\_exists='overlay' to safely add rows.
* Load Bid IDs from the Excel file into a Python set () for fast lookup and skip logic.
* Wrapped row processing in try-except blocks and logged errors without stopping the full run.

**Input and output**

Input



Output



**Conclusion: -**

This project successfully delivers a robust and automated solution for scraping bid data from the Delaware MMP Bids Portal using Python and Selenium. By covering all bid categories — Open, Recently Closed, and Not Awarded — and extracting both summary-level and modal-specific details, the scraper provides a comprehensive dataset suitable for analysis, reporting, or archiving.

Key features such as headless browsing, duplicate handling, pagination navigation, modal parsing, and structured Excel export ensure the tool is both scalable and maintainable. The modular design makes it easy to update or extend to other government procurement portals with similar structures.