# **Data Scraping Project Report**

## **Overview**

This report summarizes the approach, implementation, and results for three web scraping tasks involving financial data extraction from TradingView and SarmaayaSense websites. The goal was to collect comprehensive datasets for U.S. stock symbols, mutual funds, and detailed stock overview data..

**TASK 1: Scrape All U.S. Stock Symbols**

**Source:**

[TradingView – All U.S. Stocks](https://www.tradingview.com/markets/stocks-usa/market-movers-all-stocks/)

**Objective**:

Extract all 4,574 U.S. stock symbols along with all available attributes such as Price, Change %, Volume, Market Cap, P/E ratio, EPS, Dividend Yield, Sector, and Analyst Ratings.

**Approach:**

- Used \*\*Selenium\*\* with headless Chrome WebDriver for dynamic page interaction.

- Implemented a loop to click the \*\*"Load More"\*\* button repeatedly until all stock data is fully loaded.

- After loading all rows, extracted data from the stock table.

- Cleaned data to ensure consistency with the expected number of columns.

- Saved the final dataset into a CSV file.

**Challenges:**

- Handling dynamic content loading via the "Load More" button required waiting and exception handling to avoid missing data.

- Stale elements occasionally occurred due to DOM changes after clicks, managed with try-except blocks.

- File permission errors while saving were considered and handled gracefully.

**Outcome:**

- Successfully scraped the complete list of 4,574 stock entries.

- Data saved as a structured CSV file with 12 key attributes.

- Output file path: `/Users/farazahmed/Documents/data2.csv`

---

**TASK 2: Scrape Mutual Funds Data**

**Source:**

[SarmaayaSense – Mutual Funds](https://sarmaaya.pk/mutual-funds/)

**Objective:**

Extract data for all 508 mutual funds listed on the page, including all relevant fields available in the table.

**Approach:**

- Used \*\*Requests\*\* and \*\*BeautifulSoup\*\* for static page HTML parsing.

- Located the mutual funds data table and iterated over all rows.

- Filtered rows to ensure completeness (minimum 14 columns per row).

- Data stored into an Excel workbook using \*\*openpyxl\*\*, with automatic column width adjustment for readability.

**Challenges:**

- Ensured skipping of empty or malformed rows.

- Managed table column length discrepancies by trimming or padding data rows.

**Outcome:**

- Extracted and saved clean mutual funds data with all expected fields.

- Output Excel file path: `/Users/farazahmed/Documents/mfdata.xlsx`

- Total rows saved: 508 (matching the total mutual funds count on the website)

**TASK 3: Scrape Stock Overview Pages**

**Source:**

[TradingView – Stock Overview Example](https://www.tradingview.com/symbols/NASDAQ-AAPL/)

**Objective:**

For each of the 4,574 stock symbols obtained from Task 1, access its overview page and extract detailed fundamentals such as EPS estimate, revenue estimate, market capitalization, dividend yield, beta, and more.

**Approach:**

- Used \*\*Selenium WebDriver\*\* with Chrome (optionally headless).

- Read the stock symbols CSV file generated in Task 1.

- For each symbol:

- Constructed the URL for its overview page on TradingView.

- Waited for the fundamentals widget to load.

- Extracted fundamental metrics by locating label-value pairs on the page.

- Included robust error and timeout handling to capture any failed loads or missing data.

- Saved results into a structured CSV file.

**Challenges:**

- Variable page load times and occasional missing data fields required flexible error handling.

- Avoided detection by adding random short delays between requests.

- Managed symbols with different exchange prefixes.

**Outcome:**

- Successfully scraped detailed fundamental data for all symbols.

- Generated a comprehensive CSV with over a dozen key metrics per stock.

- Output file path: `/Users/farazahmed/Documents/task3.csv`

**Codebase and Deliverables**

- The solution is implemented in Python, utilizing Selenium for dynamic content and Requests + BeautifulSoup for static content scraping.

- Output data files are provided in CSV and Excel formats for easy consumption.

- The code is modular, with separate scripts per task and reusable helper functions (e.g., for fundamental data extraction).

- Error handling and logging are integrated to ensure smooth execution and data integrity.