**Report: Extraction and Structuring of Job Listing Data via Web Scraping**

**1. Introduction**

This report documents the process and outcomes of Task 1 of the Code Alpha Internship, which involves building a custom web scraper using Python. The objective of the task was to extract structured job listings data from an HTML page, explore HTML structure, and export the dataset for potential analysis or integration into larger data systems.

**2. Objective**

* **Building a custom web scraper** using the BeautifulSoup library to extract structured job listing data (e.g., job title, company, location, date posted, apply link) from static HTML content.
* **Understanding and navigating the structure of HTML documents** to accurately target and extract relevant information.
* **Transforming unstructured web content into a structured dataset** using pandas, enabling further exploration.
* **Performing very basic exploratory data analysis (EDA)** on the scraped dataset to draw meaningful insights about job trends, company distributions, and posting frequencies.
* **Exporting the final dataset** to a .csv file for persistence and future reuse.

This task simulated a real-world scenario where raw web content is transformed into valuable information, which helped me prepare for data-centric roles in analytics and engineering.

**3. Tools and Technologies Used**

| **Tool/Library** | **Purpose** |
| --- | --- |
| BeautifulSoup | HTML parsing and data extraction |
| requests | Sending HTTP requests to fetch web content |
| pandas | Structuring data and exporting to CSV |
| Jupyter Notebook | Interactive coding and documentation |
| Google Chrome DevTools | HTML structure inspection and element targeting |

**4. Data Extraction Process**

* Parsed a sample HTML page containing job listings using BeautifulSoup.
* Extracted the fields for each job: **Job Title, Company Name, Location, Date Posted, Apply Link**
* Stored the scraped data into a list of dictionaries and converted it into a **pandas DataFrame**.
* Exported the data to a .csv file titled fake\_job\_listings.csv.

**5. Exploratory Data Analysis (Basic Insights)**

Once the job listings were scraped and stored in a DataFrame, a basic inspection was carried out to verify the structure and integrity of the dataset:

**-** > Previewed the datasetusing `.head()` to confirm that all fields (Title, Company, Location, Date, Apply Link) were correctly extracted and aligned.

- >Confirmed that the dataset was successfully saved as a CSV (`fake\_job\_listings.csv`) with clean and readable entries.

- > Observed that the data appeared consistent across rows, with no immediate signs of structural anomalies or missing values in key fields.

- > Verified that each job listing contained all required attributes, showing the scraper was targeting the correct HTML elements.

While no deep statistical analysis was performed, this basic validation ensured the scraped dataset was complete and accurate, ready for further exploration or visualization in future steps.

**6. Key Learnings**

* Understood the fundamentals of web scraping using Python.
* Learned to navigate and interpret **HTML tags**, attributes, and nested elements.
* Gained experience in handling potential issues such as missing data, tag variations, or link extraction.
* Built a functional workflow for turning **unstructured web data into structured datasets**.

**7. Outcomes and Deliverables**

* Extracted and saved 100 structured job listings from the sample HTML.
* Successfully created fake\_job\_listings.csv for further use.
* Developed a scalable scraping logic that can be adapted for real-world applications.

**8. Conclusion**

Task 1 successfully fulfilled the internship requirements by building a real-world web scraper from scratch. The project helped me demonstrate my ability to parse HTML, extract relevant data, and transform it into a structured dataset. This task laid the foundation for scalable data gathering processes and reflected practical data engineering capabilities.