1. Website Chosen and Rationale

The website selected for this project is Books to Scrape, a well-known sandbox environment for practicing web scraping. I chose this website because I had already worked with it at the beginning of the course and was familiar with its selectors, HTML content structure, and overall format. This familiarity greatly sped up the implementation phase and allowed me to focus on deeper aspects of scraping logic and data handling. Furthermore, the site's clean layout and permissive scraping policy (robots.txt) made it a safe and compliant option for experimentation.

2. Implementation Challenges and Solutions

Challenge 1: *Parsing Incomplete Book URLs*
One notable challenge was parsing the URLs of individual book pages. The HTML source only provided partial, relative paths—such as "../../../the-black-maria_1/index.html"—which could not be used directly to access the book's page.

Solution:
To reconstruct valid URLs, I had to manually append the missing part of the path. This was done by identifying the base path (http://books.toscrape.com/catalogue/) and concatenating it with the relative URL after trimming the unnecessary "../../../" part. This step was essential to ensure the scraper could successfully access and extract information from each book's detail page.

Challenge 2: *Pagination Handling*
The website's catalog spans multiple pages, each displaying 20 books. Scraping only the landing page would result in incomplete data collection.

Solution:
The scraper was designed to detect and follow the "Next" button on each page. Using BeautifulSoup, it dynamically iterated through all pages until no

further navigation was available, ensuring that all book entries across the site were captured.

Challenge 3: *Request Timing and Stability*
Web scraping at a high frequency can overwhelm servers and result in HTTP errors or blocks.

Solution:
To mitigate this, a 2-second delay was added between requests using the time.sleep() function. Additionally, try-except blocks were employed to gracefully handle connection errors or missing pages (e.g., 404 errors), retrying where necessary to preserve continuity.

3. Analysis of Collected Data

The scraper collected metadata for approximately 1,000 books, including:

- Title
- Price
- Star Rating
- Availability
- Category
- URL

Key Observations:

- ***Price Trends:*** Most books fell in the £20–£40 range.
- ***Ratings:*** The majority had 3 or 4 stars, with a relatively low number of 5-star ratings.
- ***Availability:*** Over 80% of books were marked as "In stock".
- ***Category Distribution:*** Some categories like “Fiction” and “Science” had significantly more entries, while others such as “Poetry” had fewer.

This information was saved into CSV format and can be used for further analysis, such as comparing average prices across categories or visualizing rating distributions.

4. Potential Improvements and Extensions

Performance Enhancements:

- Replace synchronous HTTP requests with asynchronous libraries like aiohttp or a framework like Scrapy to increase speed.
- Store the scraped data in a relational database (e.g., SQLite or PostgreSQL) for more efficient querying and scalability.

Extended Functionality:

- Expand the scraper to include other mock websites (e.g., Quotes to Scrape) for multi-domain data analysis.
- Create a front-end using Flask or Tkinter for interactive exploration of the collected data.

Advanced Features:

- Add logging and unit tests to monitor scraper performance and ensure robustness.
- Automate scraping tasks using scheduling tools like cron or APScheduler to periodically update the dataset.

Deployment:

- Package the project with Docker and deploy it to a platform like Heroku or Render for remote access and real-time use.

Conclusion

This project provided hands-on experience with web scraping, emphasizing how to overcome real-world challenges like incomplete URLs and pagination. Familiarity with the website's structure allowed for quicker implementation, and the resulting dataset offers meaningful insights into online book listings. Future improvements can elevate the tool from a script-based solution to a production-ready, interactive application.