Product Scraping Documentation

1. Approach

We used **Selenium** in Python to scrape products from local e-commerce websites. Selenium allows us to control a browser and extract data from pages that load content dynamically using JavaScript.

Steps followed for each store:

1. Open the store’s search page.
2. Wait for products to load using **explicit waits**.
3. Extract product details: name, price, and link (if available).
4. Filter products based on the search keyword.
5. Combine all results into a single list.
6. Save the data to a CSV file using **pandas**.

Fallback methods were added when direct element interaction failed (e.g., using JavaScript to click or scroll).

2. Sources

We scraped the following stores:

Store Name	Website URL
Al-Fatah	https://alfatah.pk
Metro	https://www.metro-online.pk
Jalal Sons	https://jalalsons.com.pk
Carrefour	https://www.carrefour.pk/mafpak/en
Imtiaz	https://shop.imtiaz.com.pk

3. Why This Approach

- **Handles dynamic pages**: JavaScript content loads properly.
- **Easy to maintain**: Each store has its own scraping function.
- **Flexible**: Can add more stores easily in the future.
- **Data ready for use**: Products are saved in a CSV with store, name, and price.