Task 1: Product Data Extraction

Idea: The idea behind Task 1 was to automate the process of collecting key product information from ModeSens by crawling the first few pages of their collections. This allows us to build a structured dataset that includes product images and merchant availability data.

The goal of Task 1 was to extract structured product data from ModeSens, including:

- `product_id`
- `cover_url` (image of the product)
- `avail_ids` (merchant offers)
- `avail_urls` (links to purchase pages)

This process is automated through `task1.py`, which uses Selenium to navigate the first 3 pages of https://modesens.cn/collections/. For each product, it extracts the required data and saves the output in:

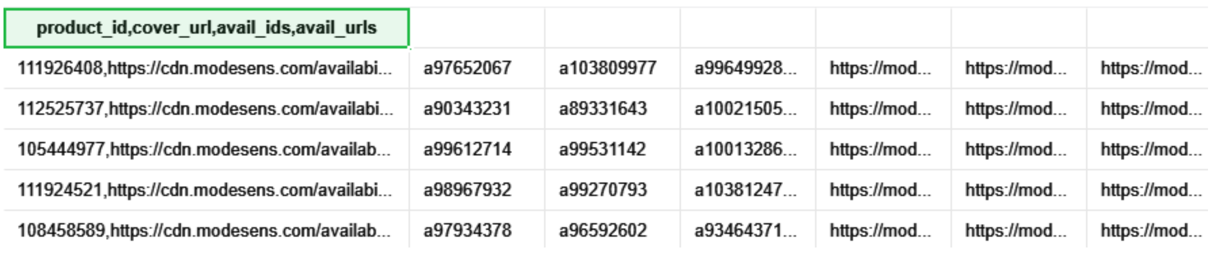
- `results/products_final.csv`
- `results/crawl.log` for tracking progress and any exceptions.

To avoid being blocked or throttled:

- Rate-limiting with `time.sleep()` is used
- The crawler respects errors and gracefully exits if the page fails or a captcha appears

If the script is interrupted or partial data is needed later, the user can run `getdata.py` to re-fetch missing info by manually providing a `product_id`. This helps recover or refine results without re-running the entire crawler.

> Note: Due to online delays and anti-bot defenses, Task 1 may not collect all products immediately. It safely handles partial scraping and allows user intervention.



product_id,cover_url,avail_ids,avail_urls						
111926408,https://cdn.modesens.com/availabi...	a97652067	a103809977	a99649928...	https://mod...	https://mod...	https://mod...
112525737,https://cdn.modesens.com/availabi...	a90343231	a89331643	a10021505...	https://mod...	https://mod...	https://mod...
105444977,https://cdn.modesens.com/availab...	a99612714	a99531142	a10013286...	https://mod...	https://mod...	https://mod...
111924521,https://cdn.modesens.com/availabi...	a98967932	a99270793	a10381247...	https://mod...	https://mod...	https://mod...
108458589,https://cdn.modesens.com/availab...	a97934378	a96592602	a93464371...	https://mod...	https://mod...	https://mod...

Task 2: Visual Similarity Matching

Idea: The goal of Task 2 was to build an intelligent, visual product exploration system. Given a product image, the user should be able to find visually similar items. This mimics the "You may also like" or "Similar items" feature seen on fashion platforms.

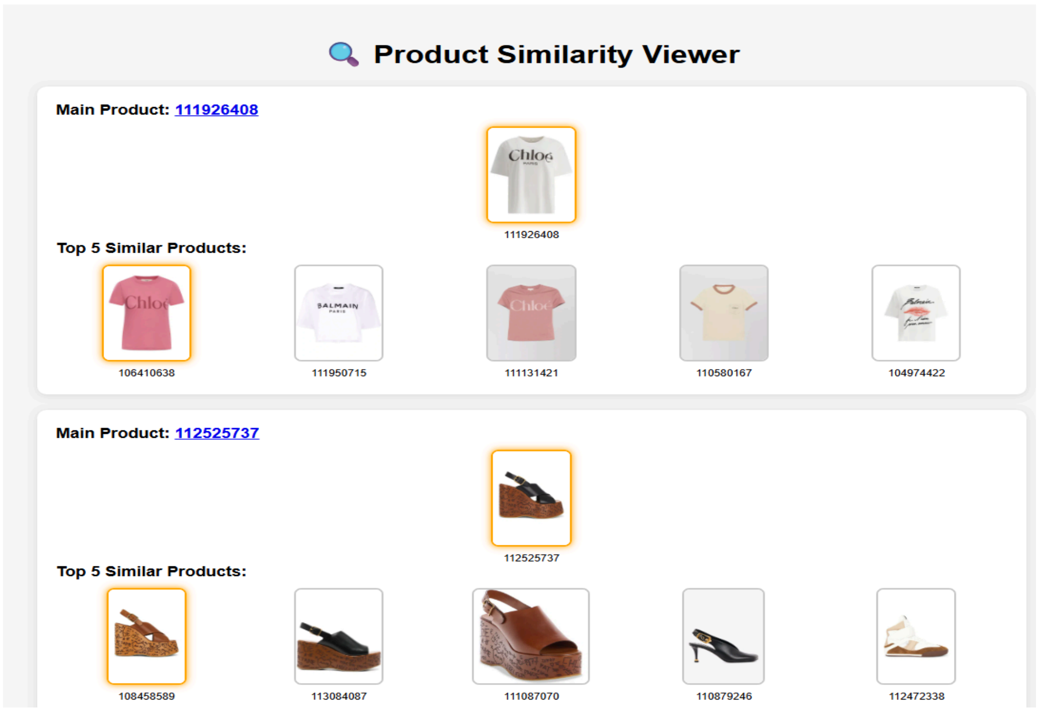
Task 2 builds an intelligent UI to visually explore similar products.

Step-by-step Process to Run Task 2:

1. **Install dependencies** (first time only):
   pip install torch torchvision scikit-learn pillow tqdm
2. **Download all product images**:
   python task2Image.py
   This saves images in the `images/` folder. Re-runs skip already-downloaded files.
3. **Run deep similarity engine** using ResNet features:
   python deep_similarity.py
   This will create `results/similarity_results.csv` with the top 5 similar products per item.
4. **Generate visual report in HTML**:
   python generate_html.py
   The output file is `similarity_report.html`, which you can open in a browser.

Task 2 Functionality Summary:

1. **Image Downloading**
   - Downloaded product `cover_url` images into a local `images/` folder.
   - Supports caching so re-running skips already-downloaded files.
2. **Visual Similarity (Multiple Versions)**
   - Initial methods used **phash** and **SSIM** for basic similarity.
   - Final solution uses **ResNet50 feature vectors** to compute deep similarity using **cosine similarity**.
   - Product 10924475 and any bad image entries are explicitly filtered out.
3. **Similarity Results**
   - Top 5 visually similar products for each item are written to `results/similarity_results.csv`
4. **HTML Report Viewer**
   - Final viewer (`similarity_report.html`) presents a styled, CSS-grid UI:
     - Main product image
     - Top 5 similar products (highlighted, linked, and labeled)
     - Uses responsive layout and hover effects for clarity


🔍 Product Similarity Viewer

Main Product: 111926408

111926408

Top 5 Similar Products:

106410638 111950715 111131421 110580167 104974422

Main Product: 112525737

112525737

Top 5 Similar Products:

108458589 113084087 111087070 110879246 112472338

🔹 Scripts Overview:

Script Name	Purpose
`task1.py`	Main product crawler (1-3 pages)
`getdata.py`	Manually fetch missing product by ID
`task2Image.py`	Downloads images
`task2_similarity_combined.py`	Hybrid phash + SSIM similarity engine
`deep_similarity.py`	Final: ResNet50-based similarity engine
`generate_html.py`	Builds final HTML similarity viewer

Output Files

- `results/products_final.csv`: Product metadata
- `results/similarity_results.csv`: Top matches
- `results/crawl.log`: Task 1 logs
- `images/*.jpg`: Local image cache
- `similarity_report.html`: Final visual output

Notes

- The system is modular: any part can be re-run separately
- Deep learning was introduced for better fashion matching
- Designed for scalability (infinite scroll supported if needed)