MILESTONE 1 REPORT: DATA ACQUISITION AND PROCESSING

Course: SEG301 – Search Engines and Information Retrieval
Project: E-Commerce Vertical Search Engine
Team: phap-bot/SEG301_Project
Completion Date: January 25, 2026

---

1 Executive Summary

The objective of Milestone 1 was to establish a reliable and scalable data foundation for a vertical e-commerce search engine. The team successfully collected over **one million product records** from seven major e-commerce platforms. A high-performance crawling system, an automated data cleaning pipeline, and a unified schema were implemented to ensure data quality and consistency.

Key Achievements

- **Total Documents:** 1,028,126 cleaned records (102.8% of the target).
- **Multi-Platform Coverage:** Data collected from 7 major e-commerce platforms.
- **Data Quality:** 99.39% retention rate after cleaning.
- **Text Processing:** 24.4 million tokens extracted for indexing.
- **System Design:** Asynchronous crawling and anti-bot mitigation strategies.

2 Team and Contributions

Name	Student ID	Role	Responsibilities
Nguyen Le Tan Phap	QE190155	Team Lead	Crawlers for Lazada, Dien May Xanh, FPTShop
To Thanh Hau	QE190039	Member	Crawlers for Tiki, Cho Tot, eBay
Nguyen Hai Nam	QE190027	Member	Crawlers for Lazada, CellphoneS

3 Data Statistics

3.1 Overall Metrics

Metric	Value
Raw Documents Crawled	1,034,466
Cleaned Documents	1,028,126
Total Tokens Extracted	24,429,834
Average Tokens per Document	23.76

3.2 Platform Distribution

Platform	Count	Percentage
Tiki	389,699	37.90%
eBay	302,083	29.38%
Cho Tot	249,146	24.23%
Lazada	34,719	3.38%
CellphoneS	31,059	3.02%
Dien May Xanh	12,140	1.18%
FPTShop	9,280	0.90%

4 Technical Architecture

4.1 Technology Stack

Languages and Runtimes

- Python 3.8+: Core crawling and data processing.
- Node.js 18+: Browser automation for JavaScript-heavy platforms.

Crawling Frameworks

- `aiohttp`, `asyncio`: High-throughput asynchronous crawling.
- Playwright, Selenium: Dynamic rendering and anti-bot handling.
- `httpx`: Asynchronous HTTP client for eBay.
- Requests: Direct API-based crawling.
- BeautifulSoup, lxml: HTML parsing and extraction.

4.2 Data Processing Pipeline

1. Data extraction using platform-specific crawlers.
2. Aggregation and global deduplication.
3. Schema normalization and missing value handling.
4. Text cleaning, tokenization, and validation.

5 Data Schema

5.1 Unified JSON Structure


```
{
  "platform": "tiki",
  "product_id": "123456789",
  "product_name": "iPhone 15 Pro Max 256GB",
  "price": 29990000,
  "original_price": 34990000,
  "discount_percent": 14,
  "product_url": "https://tiki.vn/...",
  "image_url": "https://img.tiki.vn/...",
  "rating": 4.8,
  "review_count": 1234,
  "category": "Mobile Phones",
  "segmented_text": "tiki iphone 15 pro_max 256gb mobile phones",
  "tokens": ["tiki", "123456789", "iphone","15", "pro_max", "256gb", "29990000", "349
}
```


5.2 Field Definitions

Field	Type	Description	Nullable
platform	String	Source platform identifier	No
product_id	String	Unique platform-specific ID	No
product_name	String	Product display name	No
price	Float	Current price (VND)	No
original_price	Float	Pre-discount price	Yes
discount_percent	Integer	Discount percentage	Yes
product_url	String	Product URL	No
image_url	String	Image URL	No

Field	Type	Description	Nullable
rating	Float	Average rating (0–5)	No
review_count	Integer	Number of reviews	No
category	String	Product category	No
segmented_text	String	Cleaned Vietnamese text	No
tokens	Array	Token list for indexing	No

6 Challenges and Solutions

Throughout the four-week development cycle, our team encountered several critical technical challenges. Each issue required targeted engineering solutions to ensure system stability, scalability, and data quality.

6.1 Access Blocking (Rate Limiting, HTTP 403/429)

Problem. Major e-commerce platforms such as Tiki and Ch Tt aggressively enforced rate limits, resulting in IP blocking within minutes of crawling activity.

Solution. We implemented request throttling using Python `asyncio.Semaphore` to control concurrency. Additionally, authenticated session headers, including `X-Guest-Token`, were incorporated to better emulate real user behavior and reduce bot detection.

6.2 Memory Exhaustion

Problem. Loading over one million product records into memory caused frequent system crashes due to RAM exhaustion.

Solution. The data pipeline was redesigned to use `JSONL` streaming combined with Python generators. Only product identifiers were stored in memory using a lightweight `set()`, significantly reducing memory consumption while preserving deduplication capability.

6.3 Data Inconsistency and Price Ranges

Problem. Many platforms displayed placeholder prices or broad price ranges (e.g., 30k–100k), which degraded data reliability.

Solution. We introduced a *Deep Crawl* strategy that visits individual product detail pages. This approach ensured accurate price extraction and validated inventory availability.

6.4 Lazy-Loaded Assets

Problem. Product images were frequently returned as base64 placeholders or empty strings due to lazy-loading mechanisms.

Solution. Crawlers were enhanced with explicit waiting conditions and extended DOM parsing logic to extract alternative attributes such as `data-src`, enabling reliable retrieval of actual image URLs.

6.5 Unstructured Vietnamese Text

Problem. Product titles often contained a mixture of Vietnamese, technical English terms, and informal "Teencode," complicating downstream NLP processing.

Solution. We fine-tuned `Underthesea` tokenization settings and applied custom normalization filters prior to tokenization, improving consistency and semantic quality across the dataset.

7 Conclusion

Milestone 1 successfully delivered a large-scale and high-quality dataset consisting of over one million e-commerce products. This dataset establishes a strong foundation for subsequent indexing, ranking, and search optimization stages of the project.

Report Prepared By: phap-bot Project Team
Last Updated: January 29, 2026