TUNIS BUSINESS SCHOOL
UNIVERSITY OF TUNIS

Cyber Security Project

1st deliverable

Web Scraping for Assessing Customer Satisfaction on an E-Commerce Site

Participants:

Chammaa Chaima
*ID: ****6736*
Email: chaimachammaa9@gmail.com

Jerbi Siwar
*ID: ****2645*
Email: siwar.jerbi.tbs@gmail.com

Marco Elaa
*ID: ****7089*
Email: ella.marco.tn@gmail.com

Smiri Lina
*ID: ****6262*
Email: lina01sm@gmail.com

IT-360 Information Assurance and Security
Pr. Manel Abdelkader

GitHub Repository: Security Project

Contents

1 Introduction

This project focuses on developing a **Product Review Analyzer** through **web scraping** techniques. In today's data-driven world, web scraping has become an essential tool for businesses to:

- Gather **customer feedback**
- Monitor **competition**
- Make **data-informed decisions**

Our project aims to extract and analyze product reviews from e-commerce sites to assess **customer satisfaction levels**.

2 Main Concepts and Theoretical Background

2.1 Definition of Web Scraping

Web scraping is the process of automatically extracting data from websites. It involves using software tools, scripts, or bots to retrieve and parse the content of web pages, typically in HTML format, and then converting that information into a structured format such as CSV, Excel, or a database. This technique is commonly used to gather large volumes of data from the internet for analysis, monitoring, or integration into other systems.

2.2 Functional Flow

The data collection process follows these steps:

1. Identify target **e-commerce website** and review pages
2. Analyze **HTML structure** of the pages
3. Develop **scraping script** to extract reviews
4. Store extracted data in **structured format**
5. Clean and **preprocess** the data
6. Analyze **customer satisfaction metrics**

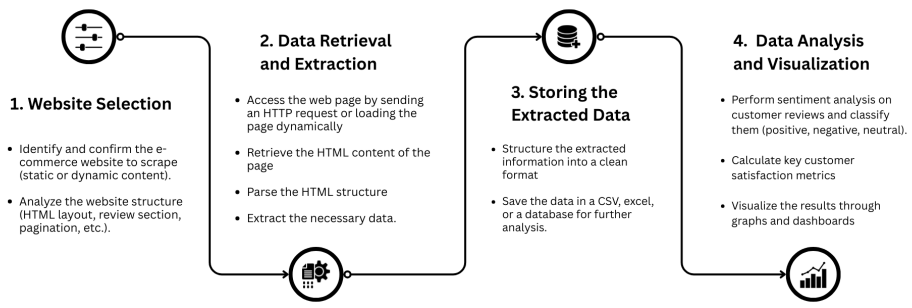

Figure 1: Functional Flow Diagram

2.3 Target Website Analysis

We first identify the structure of the target site:

- **HTML elements** (divs, spans, classes) containing reviews
- **Star rating** representations (icons, classes, numeric values)
- Compliance with the site's **robots.txt** rules

2.4 Web Scraper Engine

Our scraper will be built using **Python**. It will:

- Send **HTTP requests** using requests
- Parse **HTML** using **BeautifulSoup** or use **Selenium** for dynamic content
- Extract **relevant review** elements

2.5 Data Storage

Collected data will be stored in:

- **CSV format** for portability
- Optionally, in a **database** for easy querying and app integration

2.6 Data Cleaning & Preprocessing

We will:

- Strip unnecessary **HTML tags**
- Normalize **star ratings**
- Remove **duplicates** and irrelevant entries

2.7 Error Handling & Automation

Our script will:

- Handle **failed requests** and missing data
- Log **activities**
- Optionally support **scheduling** through automated tasks

2.8 Mathematical / Technical Background

2.8.1 XPath / CSS Selectors

XPath and **CSS Selectors** are used to locate and extract elements from HTML documents. These are essential tools in web scraping.

- **CSS Selectors**: Use patterns to select HTML elements based on tag names, classes, IDs, or element relationships
- **XPath**: A more powerful query language used to navigate XML and HTML documents

These selectors are supported by popular libraries such as **BeautifulSoup**, **lxml**, and **Scrapy**

2.8.2 Handling HTML Trees (DOM Traversal)

DOM (Document Object Model) represents the structure of HTML documents as a tree Traversal tools include:

- **BeautifulSoup**: .find(), .find_all() methods
- **lxml**: XPath support
- **Selenium**: DOM manipulation through browser automation

Traversal allows moving between parent, child, and sibling nodes to locate desired data.

2.8.3 Protocols: HTTP/HTTPS, Headers, Cookies

Web scraping relies on the **HTTP/HTTPS** protocols to send and receive data.

- **HTTP** vs. **HTTPS**: HTTPS encrypts data with **SSL/TLS** for secure transmission.
- **Headers**: Metadata sent with requests. Common headers:
  - **User-Agent**: Identifies the client software
  - **Accept-Language**: Preferred language
- **Cookies**: Store session data; some sites require cookies to maintain login sessions or track state.

Tools like **requests**, **http.client**, and browser automation tools handle these elements.

2.8.4 Robots.txt Protocol and Legal/Ethical Standards

The **robots.txt** file guides crawlers on what parts of a website can be accessed. While scraping:

- **robots.txt** should always be respected
- **Overloading** the server should be avoided
- Not extract **personal or sensitive** data

Legal/Ethical Standards:

- Many websites ban scraping in their **Terms of Service**
- Violating these terms can lead to **IP bans** or **legal actions**
- Always cite **data sources** and scrape responsibly

3 Overview of Existing Solutions

Table 1: Comparison of Web Scraping Tools

Tool	Description	Strengths	Limitations
BeautifulSoup	Simple HTML parser	**Easy to use**, flexible	Slow for **large-scale** projects
Scrapy	Full scraping framework	**Fast**, scalable	**Steep learning curve**
Selenium	Browser automation	Handles **JavaScript**	**Resource intensive**
Puppeteer	JS headless browser	Excellent **JS support**	Only in **JavaScript** environment

Table 2: Detailed Analysis of Web Scraping Solutions

Solution	Key Advantages	Major Limitations
Scrapy	• **Highly customizable** • Built for **large-scale** scraping • Integrated **pipeline** support	• Requires **Python knowledge** • Complex for **beginners**
WebHarvy	• **No coding** required • Automatic **pattern detection**	• **Windows-only** • Limited **advanced features**
Octoparse	• **User-friendly** interface • **Cloud-based** extraction	• **Less flexible** than coding • Advanced features require **payment**
Apify	• **Scalable** infrastructure • Ready-made **scrapers**	• **Usage-based pricing** • Needs **technical skills** for customization
Diffbot	• Uses **AI** for extraction • No scraper **development needed**	• **Expensive** service • Limited **control** over extraction

4 Conclusion

This document has presented our comprehensive approach to **web scraping for customer satisfaction analysis**. Key highlights include:

- Detailed examination of **existing tools** and their trade-offs
- Robust **technical framework** for implementation
- Strong emphasis on **ethical scraping practices**

Our solution aims to overcome limitations of current approaches by combining:

- **Ease of use** with powerful functionality
- **Scalability** with ethical considerations
- **Comprehensive analysis** with clear visualization

Next steps include **detailed design** and **implementation** of the scraping solution.

5 References

- **Web Scraping with Python** by Ryan Mitchell
- https://docs.python.org
- https://docs.scrapy.org
- https://developer.mozilla.org/en-US/