**MINI PROJECT-2**

***BATCH-I (TEAM NO:03)***

***PREPARED BY:PRITHIKA S A***

***COMPANY:CYBERNAUT***

***POSITION:PROJECT INTERN***

📌 **PROJECT TITLE:**

Amazon Product Scraper using Python & Selenium.

**❓ PROBLEM STATEMENT:**

🔎 Finding accurate and up-to-date product data from Amazon manually is time-consuming and inefficient.

💡 Customers, researchers, and businesses need an automated way to extract product details (names, prices, ratings, reviews, availability) for:

* 📊 Price Comparison
* 🛒 Market & consumer research
* 🏪 Competitive analysis

🎯 **OBJECTIVES:**

* 🤖 Automate the extraction of Amazon product details.
* 📂 Save results in a structured CSV format.
* -📑 Handle multiple pages of results efficiently.
* ⚙️ Allow user input for search terms, domain, pages, and output file.

**📝 PROJECT OVERVIEW:**

* This project is a Python-based Amazon scraper built with Selenium & BeautifulSoup. It automates browsing Amazon search results, extracts product details, and exports them into clean datasets.

**✨ SUPPORTS:**

* 📄 Pagination
* 👻 Headless browser
* 🔄 Deduplication
* 🌍 Custom domains

**🚀 KEY FEATURES:**

1. 🛍️ Product Extraction → ASIN, title, link, image, price, rating, reviews

2. 📄 Pagination Support → Scrape multiple result pages

3. 👻 Headless Mode → Run invisibly for efficiency

4. 🔄 Deduplication → Avoid duplicate product entries

5. 📊 CSV Export → Save structured results for analysis

**🛠️ TOOLS & LIBRARIES USED:**

* 🐍 Python 3.x
* 🌐 Selenium (browser automation)
* 🍲 BeautifulSoup (HTML parsing)
* 📊 Pandas (data storage & export)
* ⚡ WebDriver Manager (ChromeDriver setup)
* ⌨️ Argparse (command-line interface)

**CODE:**

"""

amazon\_scraper.py

Simple Selenium-based Amazon product scraper.

Works on Amazon domains (e.g., amazon.com, amazon.in).

Saves results to CSV.

"""

import time

import random

import argparse

from urllib.parse import quote\_plus, urljoin

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.chrome.options import Options

from selenium.webdriver.chrome.service import Service

from webdriver\_manager.chrome import ChromeDriverManager

from bs4 import BeautifulSoup

import pandas as pd

# ---------------------------

# Configuration / utilities

# ---------------------------

DEFAULT\_USER\_AGENT = ("Mozilla/5.0 (Windows NT 10.0; Win64; x64) "

"AppleWebKit/537.36 (KHTML, like Gecko) "

"Chrome/115.0.0.0 Safari/537.36")

def random\_sleep(a=1.0, b=2.5):

time.sleep(random.uniform(a, b))

# ---------------------------

# Browser setup

# ---------------------------

def start\_driver(headless=True, user\_agent=DEFAULT\_USER\_AGENT):

opts = Options()

if headless:

opts.add\_argument("--headless=new")

opts.add\_argument("--disable-gpu")

opts.add\_argument("--no-sandbox")

opts.add\_argument("--disable-dev-shm-usage")

opts.add\_argument(f"--user-agent={user\_agent}")

opts.add\_argument("--disable-blink-features=AutomationControlled")

# ✅ Correct way for Selenium 4+

service = Service(ChromeDriverManager().install())

driver = webdriver.Chrome(service=service, options=opts)

driver.set\_window\_size(1200, 900)

return driver

# ---------------------------

# Parsing helpers

# ---------------------------

def parse\_search\_results(html, base\_domain):

soup = BeautifulSoup(html, "html.parser")

products = []

results = soup.select('div[data-asin][data-component-type="s-search-result"]')

for r in results:

asin = r.get("data-asin", "").strip()

if not asin:

continue

# Title & link

title\_tag = r.select\_one("h2 a.a-link-normal.a-text-normal") or r.select\_one("h2 a")

title = title\_tag.get\_text(strip=True) if title\_tag else ""

link = urljoin(base\_domain, title\_tag.get("href")) if title\_tag and title\_tag.get("href") else ""

# Image

img = r.select\_one("img.s-image")

img\_url = img.get("src") if img else ""

# Price

price\_whole = r.select\_one(".a-price .a-price-whole")

price\_frac = r.select\_one(".a-price .a-price-fraction")

if price\_whole:

price = price\_whole.get\_text(strip=True) + (("." + price\_frac.get\_text(strip=True)) if price\_frac else "")

else:

price\_span = r.select\_one("span.a-offscreen")

price = price\_span.get\_text(strip=True) if price\_span else ""

# Rating & reviews

rating\_tag = r.select\_one(".a-icon-alt")

rating = rating\_tag.get\_text(strip=True) if rating\_tag else ""

reviews\_tag = r.select\_one("span.a-size-base")

reviews = reviews\_tag.get\_text(strip=True) if reviews\_tag else ""

products.append({

"asin": asin,

"title": title,

"link": link,

"image": img\_url,

"price": price,

"rating": rating,

"reviews": reviews

})

return products

# ---------------------------

# Scraper main

# ---------------------------

def scrape\_amazon(search\_term, pages=1, domain="https://www.amazon.in", headless=True, delay=(2.0,4.0)):

driver = start\_driver(headless=headless)

all\_products = []

try:

for page in range(1, pages+1):

q = quote\_plus(search\_term)

search\_url = f"{domain}/s?k={q}&page={page}"

print(f"[+] Loading page {page}: {search\_url}")

driver.get(search\_url)

random\_sleep(delay[0], delay[1])

driver.execute\_script("window.scrollTo(0, document.body.scrollHeight/3);")

random\_sleep(0.5, 1.2)

driver.execute\_script("window.scrollTo(0, document.body.scrollHeight);")

random\_sleep(0.5, 1.0)

html = driver.page\_source

products = parse\_search\_results(html, domain)

print(f" -> Found {len(products)} results on page {page}")

all\_products.extend(products)

random\_sleep(delay[0]+0.5, delay[1]+1.0)

finally:

driver.quit()

# Deduplicate

seen = set()

dedup = []

for p in all\_products:

if p["asin"] and p["asin"] not in seen:

dedup.append(p)

seen.add(p["asin"])

return dedup

# ---------------------------

# CLI / Export

# ---------------------------

def save\_to\_csv(items, filename="amazon\_results.csv"):

df = pd.DataFrame(items)

df.to\_csv(filename, index=False)

print(f"[+] Saved {len(items)} items to {filename}")

def main():

parser = argparse.ArgumentParser(description="Amazon search scraper (Selenium)")

parser.add\_argument("query", help="Search query text (use quotes if spaces)")

parser.add\_argument("--pages", "-p", type=int, default=1, help="Number of search result pages to scrape")

parser.add\_argument("--domain", "-d", default="https://www.amazon.in", help="Amazon base domain")

parser.add\_argument("--headless", action="store\_true", help="Run Chrome headless")

parser.add\_argument("--out", "-o", default="amazon\_results.csv", help="Output CSV filename")

args = parser.parse\_args()

items = scrape\_amazon(args.query, pages=args.pages, domain=args.domain, headless=args.headless, delay=(2.0,4.0))

save\_to\_csv(items, filename=args.out)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**▶️ HOW TO RUN:**

1️⃣ Install Dependencies:

Pip install selenium beautifulsoup4 pandas webdriver-manager

2️⃣ Run the Scraper:

Python amazon\_scraper.py “laptop” –pages 2 –domain <https://www.amazon.in> –out laptops.csv –headless

3️⃣ Command-Line Options:

* 🔍 query → Search query (e.g., “laptop”)
* 📄 –pages, -p → Pages to scrape (default: 1)
* 🌍 –domain, -d → Amazon domain (default: in)
* 💾 –out, -o → Output CSV filename (default: amazon\_results.csv)
* 👻 –headless → Run browser in headless mode

**OUTPUT STRUCTURE**:

Scraped data is saved as CSV with columns:

* 🆔 asin
* 🏷️ title
* 🔗 link
* 🖼️ image
* 💰 price
* ⭐ rating
* 🗣️ reviews

B0C123XYZ HP Laptop 15s https://... https://... ₹45,990 ⭐4.2 1,234

B09XYZ789 Dell Inspiron 3520 https://... https://... ₹52,490 ⭐4.3 2,450

💡 **EXAMPLE USE CASES:**

* 🛍️ Price comparison apps
* 📊 Market research & analytics
* 🎓 Academic projects
* 🏪 Competitive seller analysis