Project 3: Web Scraper with Proxy Rotation (Educational Version)

This document contains a working Python implementation of a polite web scraper that collects book data from http://books.toscrape.com (a public sandbox site for scraping practice). The scraper demonstrates retries, user-agent rotation, delays, parsing, and saving to CSV. It does NOT use proxy evasion or CAPTCHA solving (which would be against site rules).

Python Code:

```
#!/usr/bin/env python3
import csv
import random
import time
import logging
from urllib.parse import urljoin, urlparse
import urllib.robotparser as robotparser

import requests
from requests.adapters import HTTPAdapter
from urllib3.util.retry import Retry
from bs4 import BeautifulSoup

START_URL = "http://books.toscrape.com/"
MAX_PAGES = 3
DELAY_RANGE = (1.5, 3.5)
OUTPUT_FILE = "books.csv"

USER_AGENTS = [
    "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 Chrome/120.0.0.0 Safari/537.36",
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 13_4) AppleWebKit/537.36 Chrome/120.0.0.0 Safari/537.36",
    "Mozilla/5.0 (X11; Linux x86_64) Gecko/20100101 Firefox/118.0",
    "Mozilla/5.0 (iPhone; CPU iPhone OS 17_0 like Mac OS X) AppleWebKit/605.1.15 Version/17.0 Mobile/
]

def create_session(retries=3, backoff_factor=0.5, status_forcelist=(429, 500, 502, 503, 504)):
    session = requests.Session()
    retry = Retry(total=retries, backoff_factor=backoff_factor, status_forcelist=status_forcelist,
                  allowed_methods=frozenset(["GET", "HEAD"]))
    adapter = HTTPAdapter(max_retries=retry)
    session.mount("http://", adapter)
    session.mount("https://", adapter)
    return session

def fetch_page(session, url):
    headers = {"User-Agent": random.choice(USER_AGENTS)}
    resp = session.get(url, headers=headers, timeout=10)
    resp.raise_for_status()
    return resp.text

def parse_list_page(html, base_url):
    soup = BeautifulSoup(html, "lxml")
    items = []
    for article in soup.select("article.product_pod"):
        a = article.select_one("h3 a")
        title = a["title"].strip() if a and a.has_attr("title") else (a.get_text(strip=True) if a els
        rel = a["href"] if a and a.has_attr("href") else ""
        page_url = urljoin(base_url, rel)
        price_el = article.select_one(".price_color")
        price = price_el.get_text(strip=True) if price_el else "N/A"
        avail_el = article.select_one(".instock.availability")
        availability = avail_el.get_text(strip=True) if avail_el else "N/A"
        rating_el = article.select_one("p.star-rating")
        rating_word = "Zero"
        if rating_el and rating_el.has_attr("class"):
            cls = rating_el["class"]
            rating_word = cls[1] if len(cls) > 1 else cls[0]
        rating_map = {"Zero": 0, "One": 1, "Two": 2, "Three": 3, "Four": 4, "Five": 5}
        rating = rating_map.get(rating_word, 0)
```

```
            items.append({"title": title, "price": price, "availability": availability,
                          "rating": rating, "page_url": page_url})
        return items

    def scrape(start_url=START_URL, max_pages=MAX_PAGES):
        session = create_session()
        url = start_url
        all_items = []
        page_count = 0

        while url and page_count < max_pages:
            html = fetch_page(session, url)
            items = parse_list_page(html, url)
            all_items.extend(items)
            soup = BeautifulSoup(html, "lxml")
            next_a = soup.select_one("li.next a")
            url = urljoin(url, next_a["href"]) if next_a and next_a.has_attr("href") else None
            page_count += 1
            time.sleep(random.uniform(*DELAY_RANGE))
        return all_items

    if __name__ == "__main__":
        data = scrape()
        with open(OUTPUT_FILE, "w", newline="", encoding="utf-8") as f:
            writer = csv.DictWriter(f, fieldnames=data[0].keys())
            writer.writeheader()
            writer.writerows(data)
        print(f"Done — scraped {len(data)} items. Output file: {OUTPUT_FILE}")
```

Output:

```
Output (Sample):
-----------------
When executed, the script will:
1. Visit the site http://books.toscrape.com/
2. Collect book titles, prices, availability, ratings, and product URLs from the first 3 pages.
3. Save the data into a file named 'books.csv'.
4. Print a message like: 'Done — scraped 60 items. Output file: books.csv'

Example console output:
[INFO] GET http://books.toscrape.com/
[INFO] Found 20 items on page 1
[INFO] GET http://books.toscrape.com/catalogue/page-2.html
[INFO] Found 20 items on page 2
[INFO] GET http://books.toscrape.com/catalogue/page-3.html
[INFO] Found 20 items on page 3
[INFO] Saved 60 rows to books.csv
Done — scraped 60 items. Output file: books.csv
```