Розроблена програма дозволяє отримувати опис, коментарі інших людей та оцінку будь-якого електронного пристрою. Виконання програми не призводить до збудження помилок в програмі та на виході завжди повертає результат незалежно від вхідних даних, а саме назви шуканого пристрою. Цим результатом може бути оцінка нуль та відсутність опису та коментарів, в разі некоректно введеної назви чи просто як наслідок пошуку недостатньо відомого товару, для якого не залишено жодного відгуку на всіх трьох інтернет ресурсах. В такому разі користувачу буде повідомлятися про те, що в результаті пошуку інформації про введений пристрій не було знайдено жодних даних.

Для видобування даних використовувалися три модулі кожний з яких відповідав за конкретний інтернет ресурс. Ідея була такою, що кожний цей модуль має функцію котра приймає назву продукту та об’єкт класу Comments і робить певні зміни у цьому об’єкті, а саме добавляє до списку, який це об’єкт інкапсулює нові об’єкти класу Comment. Це дозволило під час пошуку всіх коментарів створити один об’єкт класу Comments та просто по черзі викликати три функції, передавши у кожну з них цей об’єкт. По завершенню виконання цих функцій ми отримали “заповнений” об’єкт, з якого ми можемо отримати середню оцінку і всі коментарі.

Для того, щоб реалізувати ідею з поступовою підгрузкою інформації, використовуючи JavaScript та Ajax запити, нам довелося створювати три об’єкти класу Comments та в кожний з них добавляти коментарі з різних джерел. Після повернення збережених у об’єкті даних посилання на цей об’єкт пропадало, а отже об’єкт видалявся за допомогою garbage collector. З цього слідує висновок, що обсяг використаної пам’яті не збільшився.

Приклад модуля, котрий добавляє у Comments коментарі з “Rozetka”.

# File: rozetka\_parse.py  
# Module has a functionality that allow take comments from rozetka web shop.  
from bs4 import BeautifulSoup  
import urllib.request  
from comment import Comment  
  
  
def insert\_comments\_1(container, name\_of\_product):  
 *""" Create comments from rozetka web shop and add them to container. """* url = "https://rozetka.com.ua/ua/search/?text=" + name\_of\_product  
 try:  
 with urllib.request.urlopen(url) as response:  
 html = response.read().decode("utf-8")  
 soup = BeautifulSoup(html, "html.parser")  
 new\_url = soup.find("div", class\_="g-i-tile-i-box-desc").find("a")[  
 "href"] + "/comments/sort=helpful/"  
 for com in get\_comments(new\_url):  
 mark = com.find("span", "sprite g-rating-stars-i")  
 if mark is not None:  
 mark = int(mark.get("content"))  
 content, text = com.findAll("div", "pp-review-text-i"), ""  
 for i in range(len(content)):  
 text += content[i].text.strip() + "\n"  
 text = text.strip()  
 if text[-1] != "?":  
 container.commants.append(Comment(text, mark))  
 except AttributeError:  
 pass  
  
  
def get\_comments(url):  
 *""" str -> str """* with urllib.request.urlopen(url) as response:  
 soup = BeautifulSoup(response.read().decode("utf-8"), "html.parser")  
  
 return soup.find\_all("div", "pp-review-inner")

Важливим є те, що в результаті виконання функції insert\_comment\_1 (принцип роботи якої є ідентичним до принципів роботи двох інших вище згаданих функцій), не може збуджуватися помилка, котра б не обробилася у цій самій функції. При виникненні помилки функція просто повертає None, та залишає об’єкт незмінним. Якщо помилка виникає вже після добавлення деяких коментарів, то функція ідентично повертає None, однак об’єкт Comments вже зберігає у собі видобуванні коментарі.

Очевидно, що якщо помилка й збудиться то це буде Attribute Error, тому що кожна з використовуваних функцій повертає або об’єкт, котрий потрібен програмі або None. В першому випадку продовжить коректне виконання, в іншому випадку збудиться Attribute Error.

Висновки

Під час виконання курсової роботи виникла проблема видобування інформації з сторінок, інформація на яких оновлюється за допомогою JavaScript одразу після їхнього завантаження. Використовуючи модуль urllib.requests чи requests для зчитування веб сторінки, такі сторінки зчитуються не повністю, без інформації, яку мав добавити JavaScript.

Краще використовувати просту структуру даних, якщо вона повністю може надати бажаний інтерфейс та логіку, і не придумувати “велосипед”. Після спрощення абстрактних типів даних логіка нашої програми стала значно простішою та елегантнішою. Незважаючи на це, результати роботи практично не відрізняються.

Під час роботи з проектом ми вивчили багато нових технологій, таких як Django, JavaScript, Flask, Sass, Html, CSS, Docker, Docker-Compose, PostgreSQL, SQLite 3, Ajax та Beautiful Soup.

Також здобули досвід командної роботи.