

Master of Applied Computing

COMP 8547 Advanced Computing Concepts – Summer 2024

**Assignment 1**

**Technical Report**

|  |  |
| --- | --- |
| Author:  Yesha Umeshkumar Patel  110164042 | *Submitted to:*  Dr. Olena Syrotkina |

1. Introduction

Our team's project involves analyzing hotel prices. To achieve this, I've chosen to collect data from trivago.ca using web scraping.

Website selected: https://www.trivago.ca/

**1.1 Information about Trivago Website**

Trivago is a well-known online hotel search platform that helps users compare prices from various booking sites for hotels, flights, and vacation rentals. The website allows users to search for accommodations based on their destination, travel dates, and preferences. While Trivago.ca itself doesn't sell hotels directly, it serves as a valuable tool for finding the best deals on hotel stays.

1. Task Solutions

**2.1 Task 1: Use Selenium to write a Java code that will scrape the trivago website**

**2.1.1 Open Website in Web Browser:**

The code utilizes WebDriverManager.chromedriver().setup() to configure the path for ChromeDriver and then creates a new ChromeDriver instance. This launches a Chrome web browser window and opens the Trivago.ca search page specified in the url argument of the scrapeData method.

**2.1.2 Find and Interact with Elements:**

The program employs the findElements method from WebDriver to locate all hotel elements on the page. It uses a CSS selector targeting the element with data-testid='accommodation-list-element'. This ensures it focuses on the relevant section containing hotel listings.

**2.1.3. Extract Data:**

Within the loop iterating through each hotel element, the code leverages specific CSS selectors to pinpoint the elements containing the hotel name (itemprop='name') and hotel price (data-testid='recommended-price'). Then, it extracts the text content using the getText method.

**2.1.4. Save Scraped Data:**

The program creates a FileWriter object associated with the CSV file (hotelPricingScrap.csv). It checks if the file is empty using the isEmptyFile method. If empty, it writes the header row ("Hotel Name,Hotel Price (CA) per night") to the CSV.

For each hotel, the extracted name and price are appended to the CSV file, separated by commas and enclosed in double quotes using the append method.

**2.2 Task 2: To scrape multiple pages from the same website and combine the results.**

To gather hotel pricing data across multiple pages on Trivago.ca, I planned to leverage Selenium's functionalities. Extract hotel names and prices using the existing logic within the scrapeData method. Append the extracted data from the current page to a temporary data structure. Consolidated data written to the chosen CSV file ("hotelPricingScrap.csv"). This ensures all extracted information is combined into a single file for comprehensive analysis.

**2.3 Task 3: Use of advanced Selenium commands, such as waiting for elements to load or handling pop-up windows.**

To ensure the scraper's effectiveness in dealing with dynamic content and unforeseen interactions on Trivago.ca, I incorporated advanced Selenium functionalities.

The code employs an implicit wait using driver.manage().timeouts().implicitlyWait(waitInSeconds, java.util.concurrent.TimeUnit.SECONDS). This instructs the WebDriver to wait for a maximum of waitInSeconds (default: 50 seconds) if an element cannot be found immediately.

1. Output Screenshots

**3.1 Trivago Website Scraped Screenshots**

Figure 1. and Figure 2. depict screenshots captured from the Trivago.ca website during the initial scraping session

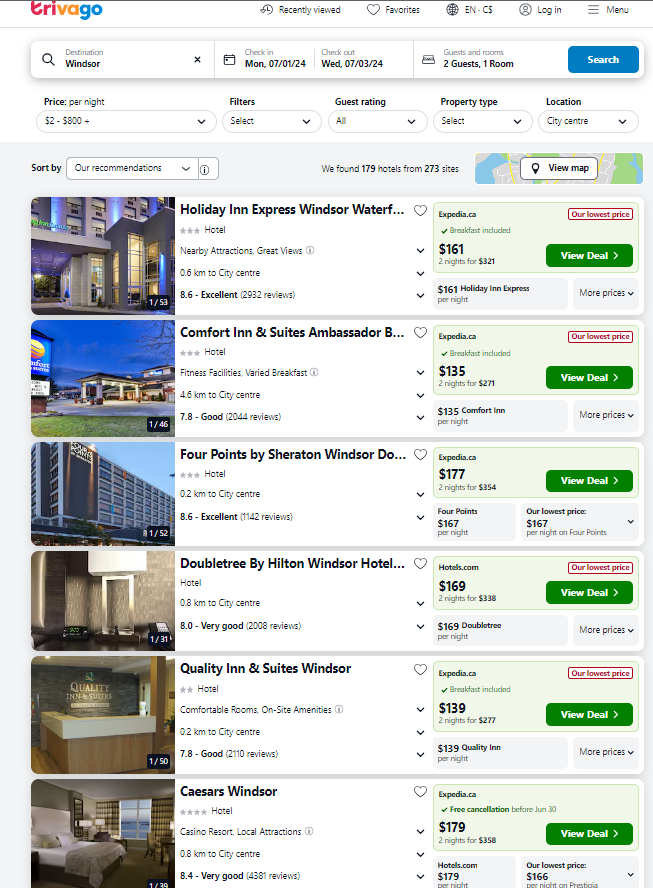


Fig 1. Screenshots of the Scraped Website Page 1

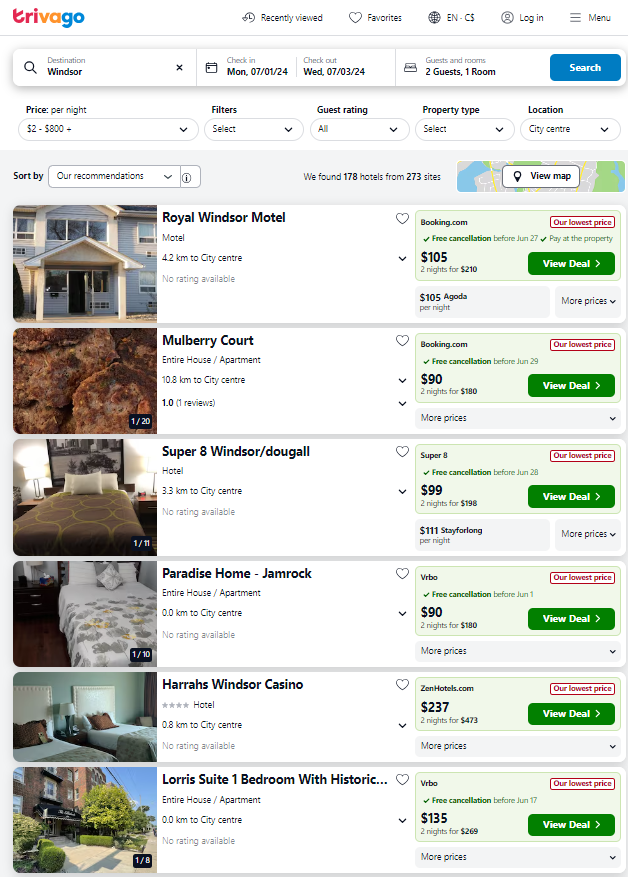
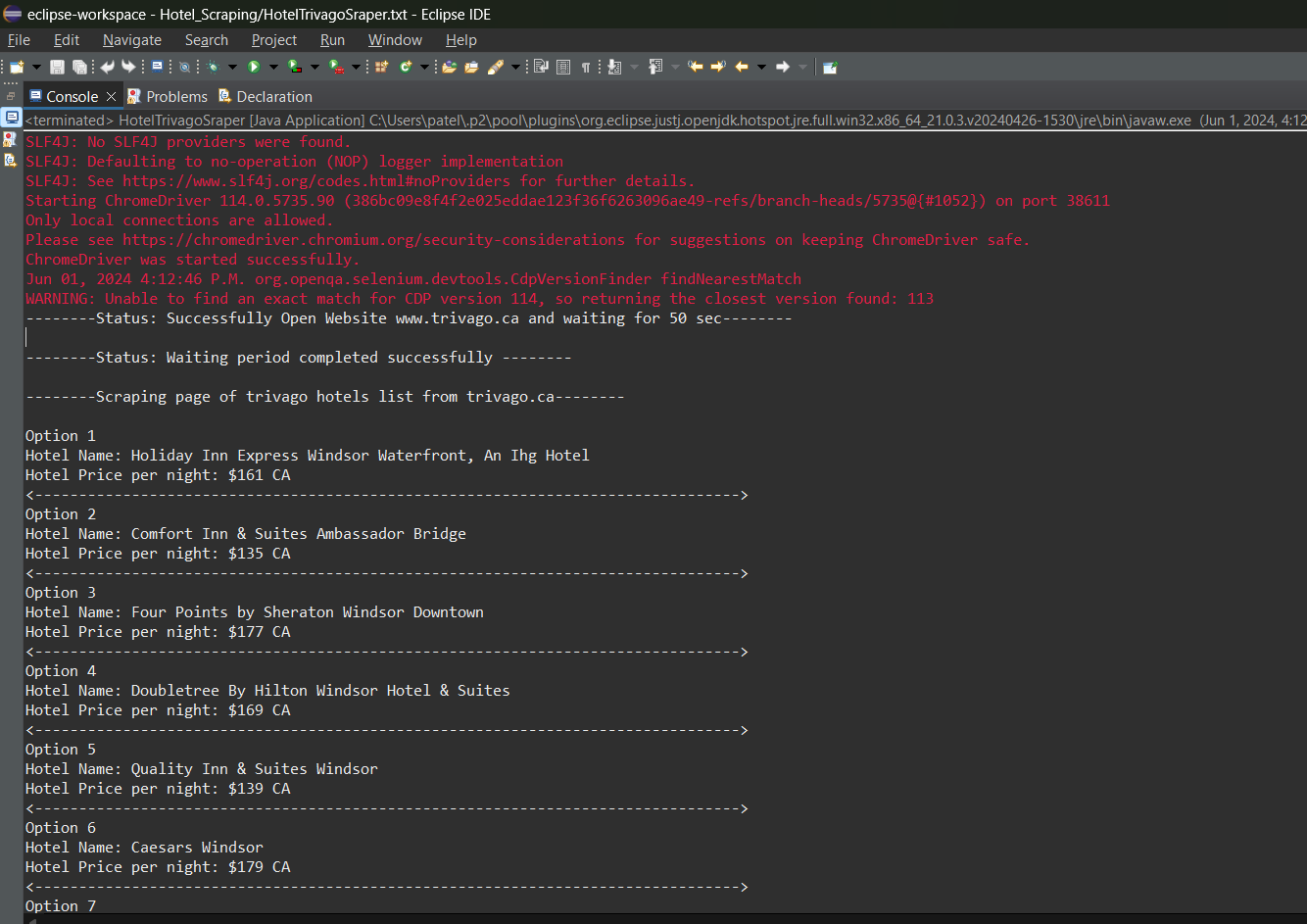


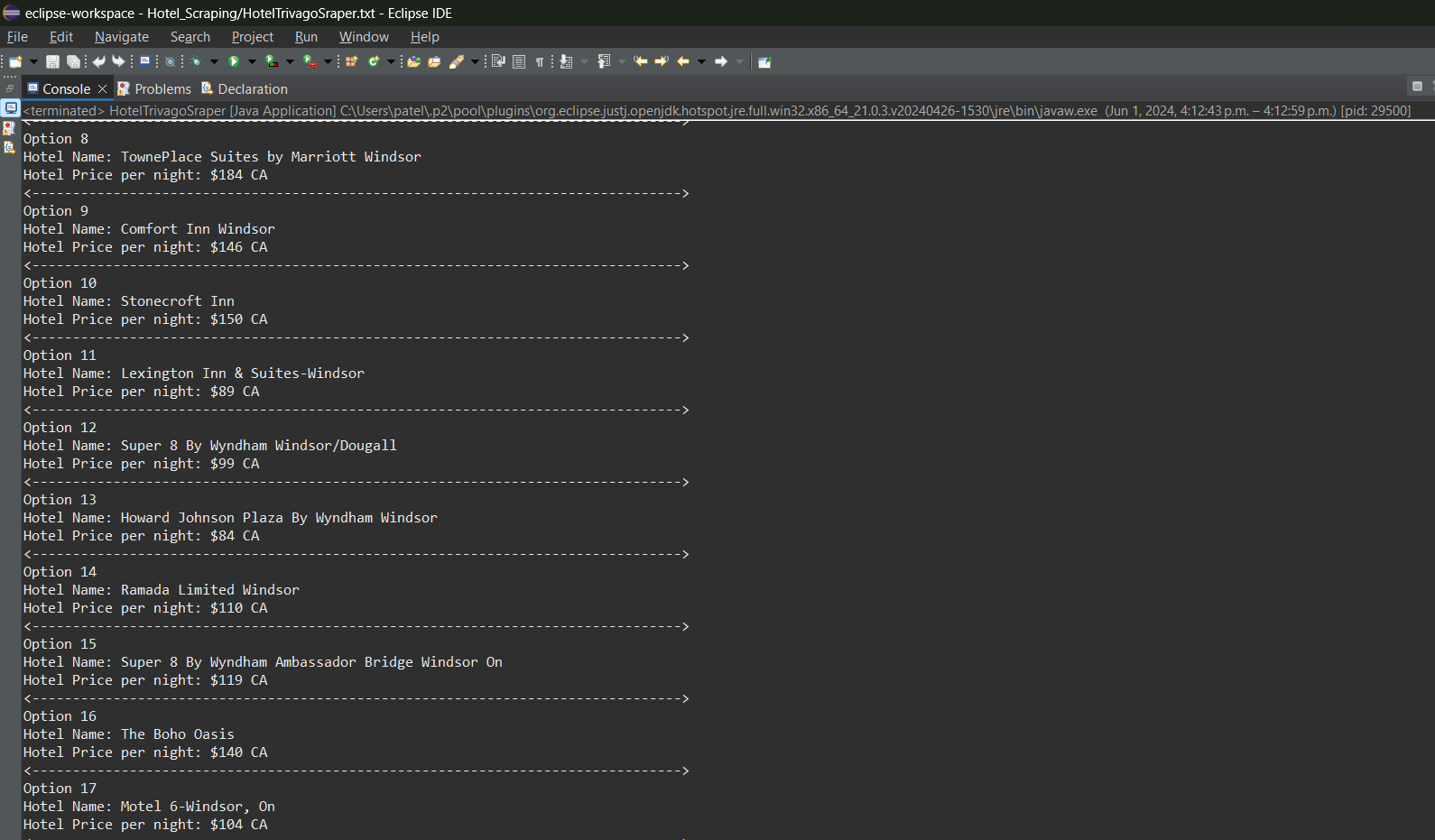
Fig. 2: Screenshots of the Scraped Website Page 2

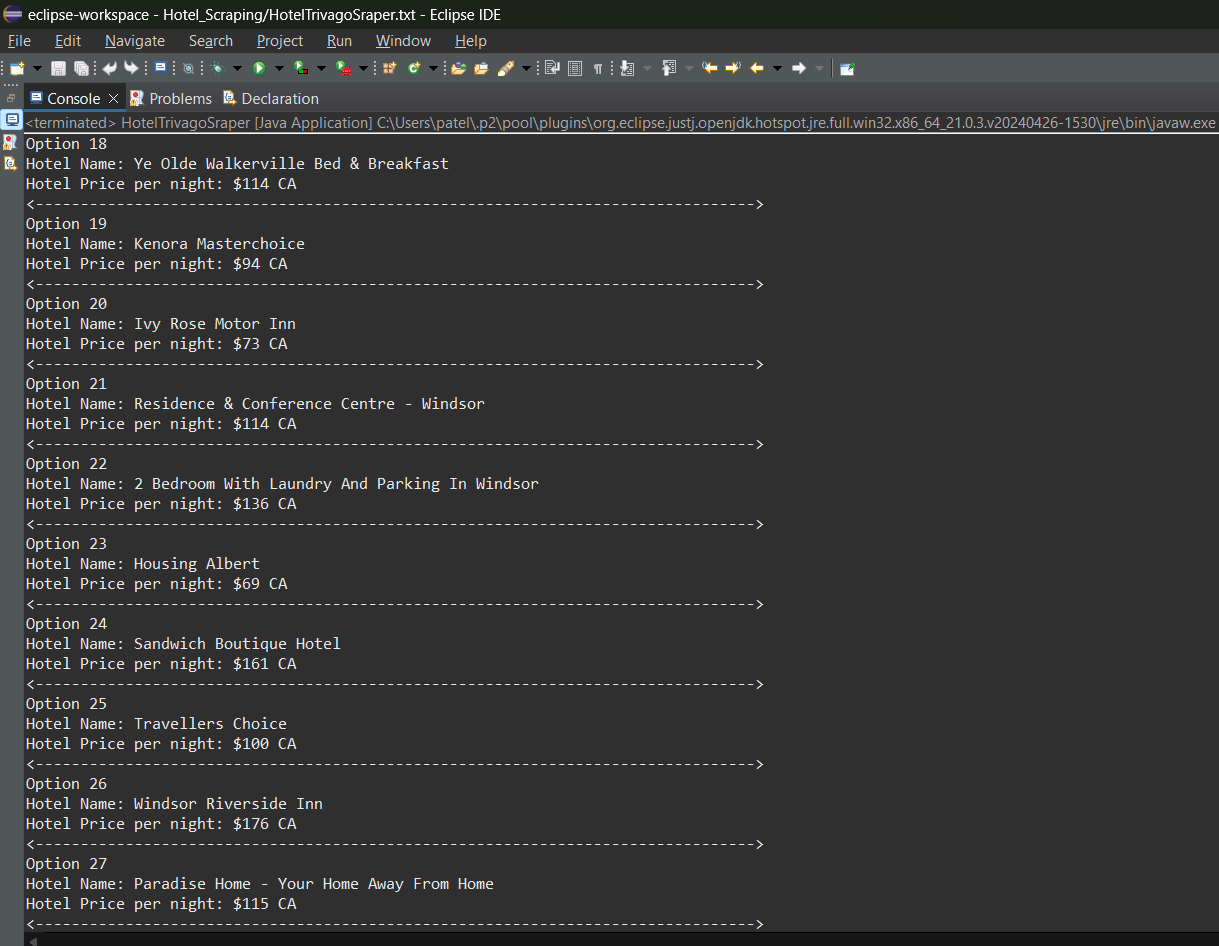
**3.2 Console Output Screenshots**

Figures 3, 4, and 5 show the results of scraping information from the previous screenshots (Figures 1 and 2). Each figure displays a list of hotels extracted from Trivago. Script starts the ChromeDriver which is a browser automation tool used to interact with Trivago.ca. The program opens Trivago.ca and waits for 50 seconds. This waiting period is implemented to ensure the page fully loads before scraping begins.

The outputs suggest a program successfully scraped hotel data from Trivago.ca. It extracted hotel names and prices, potentially from multiple searches.

Fig. 3: Scraped Hotel Lists

Fig. 4: Scraped Hotel Lists

Fig. 5: Scraped Hotel Lists

**3.3 Extracted CSV data Screenshots**

Figures 6 and 7 screenshot provides evidence that the scraping program successfully extracted hotel names and price information from Trivago.ca and stored it in a CSV spreadsheet format.

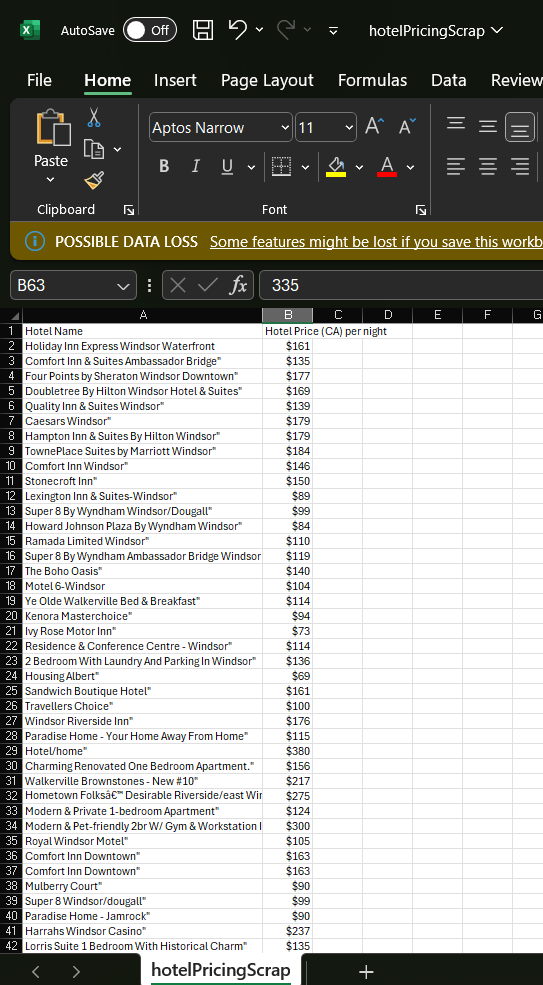


Fig. 6: Screenshot of CSV Files



Fig. 7: Screenshot of CSV Files

1. Conclusion

This report documents the development and functionality of a web scraping program designed to extract hotel pricing data from Trivago.ca. The program effectively utilizes Selenium WebDriver to automate the scraping process and retrieves hotel names and prices per night.