DFA TEKNOLOJİ

WEB BASED DATA COLLECTION AND PRICE COMPARISON TOOL FINAL REPORT

by

Betül Ulutürk

IZMIR

23.05.2025

CHAPTER ONE

PROGRESS DESCRIPTION

The development of this project started with analysis HTML structure of product pages on Hepsiburad. I noticed that some information is visible right away, the list of other sellers only appears after clicking the "Other Sellers" button. That means request wouldn’t be enough. So I use Selenium, which allowed me to automate a browser and load the full content dynamically.

This project involves the development of a Python application that collects and analyzes pricing and other seller information for a given product on the Hepsiburada platform. Based on a product page URL provided by the user, the application utilizes Selenium and BeautifulSoup libraries to analyze the page content and perform comparison and analysis among different sellers.

CHAPTER TWO

PROBLEM DEFINITION

On multi-seller platforms like Hepsiburada, different price, shipping and seller rating information is provided for the same product. Users have to compare all this information manually. The tool developed to solve this problem lists and analyzes sellers according to the product link entered by the user. The system shows the 5 most expensive and cheapest sellers of the product to the user along with their information. It also saves all seller information in a csv file with their information.

CHAPTER THREE

METHODS AND TOOLS USED

While developing this project, I use Python programming language. Seller information is not immediately available upon page load, but is dynamically rendered after clicking the "Other Sellers" button. Therefore, using the requests library alone was not enough fort his action. So I use Selenium to interact with browser and load dynamic content.

Once the full page source was obtained, sellers data was extracted using BeautifulSoup and organized format with pandas. Data exported as a CSV file and analyzed the top 5 cheapest and most expensive sellers, which were presented to the user by a simple GUI.

During development, several challenges were encountered in the HTML. As a result, a fully functional desktop application was created, capable of taking a Hepsiburada product URL as input, analyzing seller data, and displaying the results interactively.

I also wanted to make the tool easy to use for someone who doesn't code, so I added a graphical interface using Tkinter. That way, users can easily paste a product URL into a box, click the button, and see the results of comparisons. I even used the Pillow library to display a little project logo in the interface, just to give it a more good impression.

CHAPTER FOUR

PROBLEMS ENCOUNTERED AND SOLUTIONS

One of the biggest challenges I faced during this project was working with dynamically loaded content on the Hepsiburada product pages. At first, I tried using simple HTML parsing tools like requests and BeautifulSoup. But I realized that the seller section wouldn’t load unless a other sellers button was clicked on the page. So I use Selenium, which allowed me to interact with the page just like a user would click button, waiting for elements, and then collecting the full content that I wanted.

I also have an issue where the user input (URL) wasn’t always valid. For instance, users might paste a general search result page or a non-Hepsiburada link. To fix this, I added basic validation to make sure the URL starts with the correct pattern and contains product-specific content.

These problems really helped me get more comfortable with messy, real-world data. It also taught me how small improvements in code and design can make a big difference in how smooth and usable an app feels.

CHAPTER FIVE

CONCLUSION AND EVALUATION

This project started with a simple goal: to help users easily compare prices and seller information on Hepsiburada without having to dig through the site themselves. In the end, the app does exactly that it takes a product link, gathers all the seller data, and presents it in a clear, easy to read way.

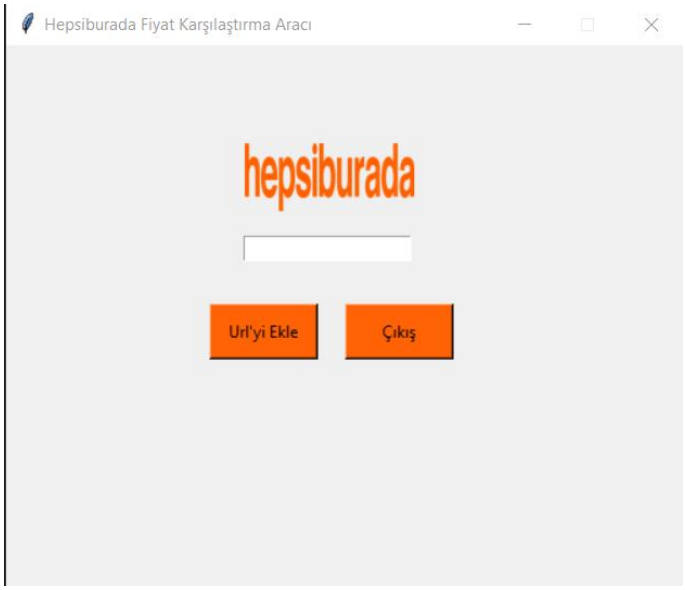
Using Selenium and BeautifulSoup together good for dealing with Hepsiburada's dynamic content. And with the help of Tkinter, I was able to build a small GUI that makes the whole process feel more approachable even for people who don't write code.

More than just getting it to work, this project taught me a lot about building something practical from scratch. It showed me how different tools can come together to solve a real problem and how much of a difference a simple, user-friendly interface can make. I learn tkinter thanks to this Project and I gained experience in developing and using my Python skills.

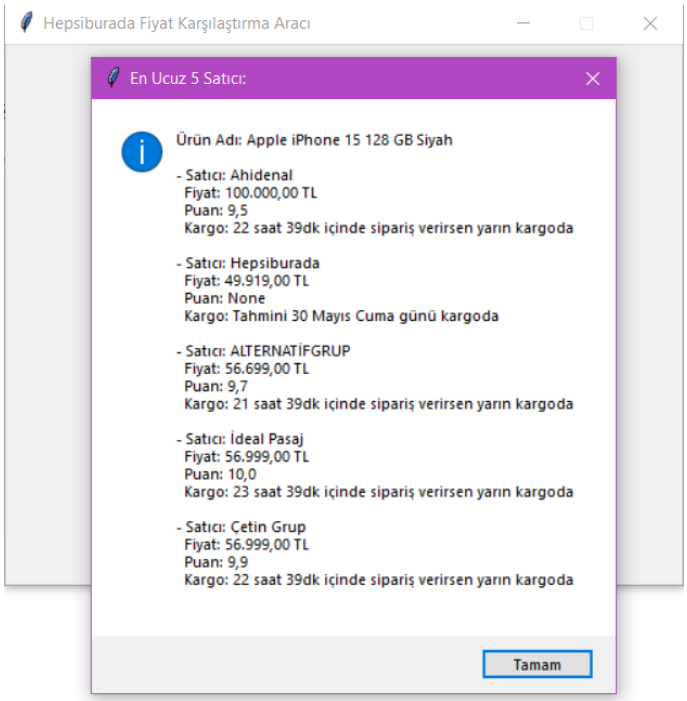
CHAPTER SIX

SCREENSHOTS

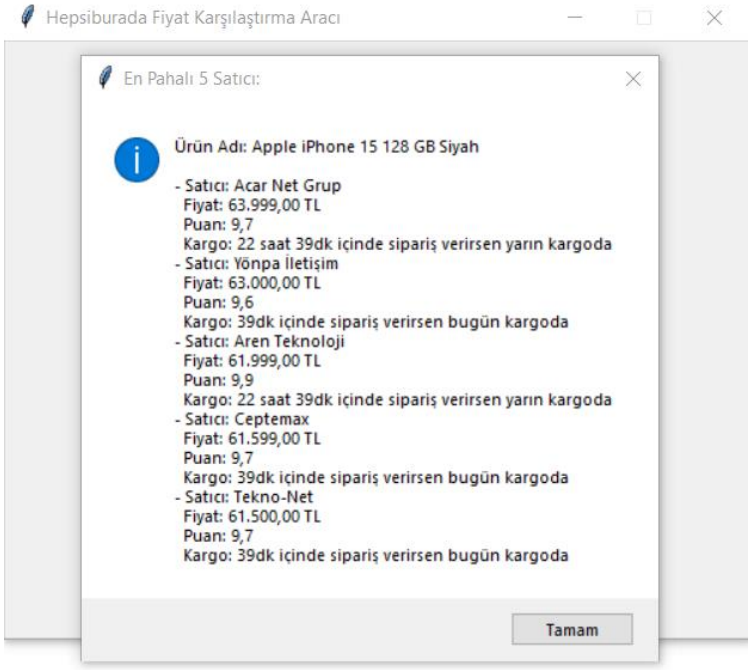
MAIN SCREEN:



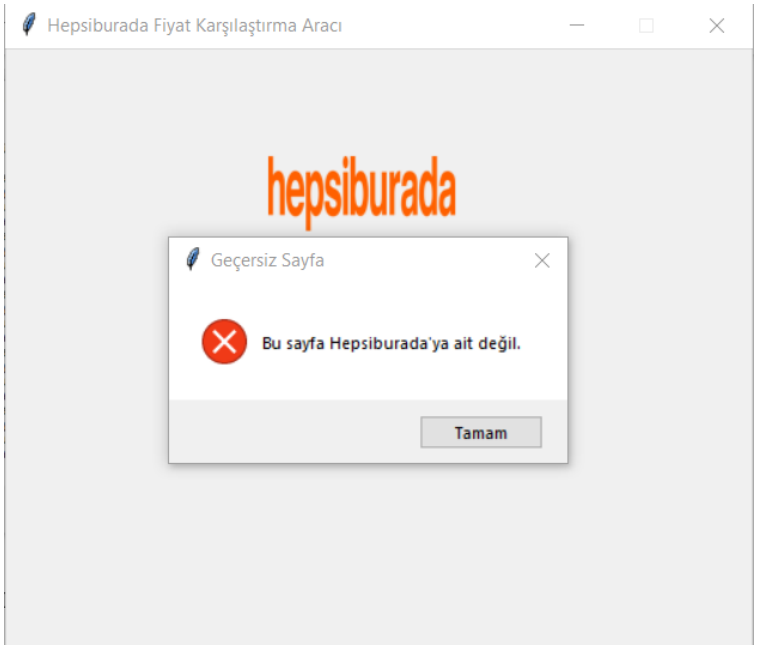
SHOWING 5 CHEAPEST SELLERS:



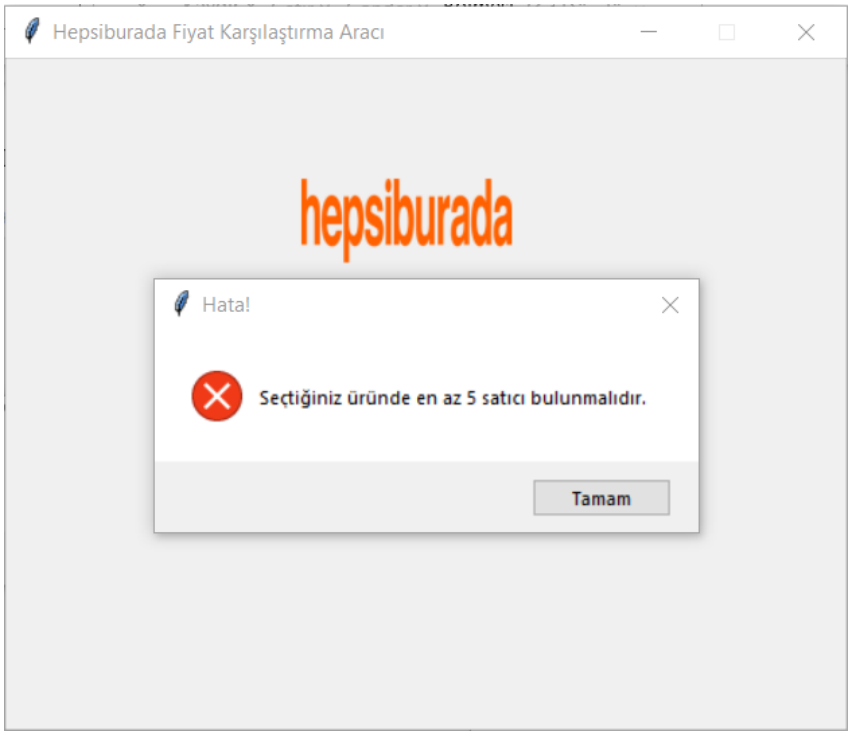
SHOWING 5 MOST EXPENSIVE SELLERS:



WHEN ADDING WRONG URL:



WHEN THERE ARE FEWER THAN 5 SELLERS:



REFERENCES

[1] https://www.selenium.dev/documentation

[2] https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[3] https://docs.python.org/3/

[4] https://pandas.pydata.org/docs/

[5] https://pypi.org/project/webdriver-manager/

[6] https://docs.python.org/3/library/tkinter.html

[7] https://pillow.readthedocs.io/en/stable/

[8] https://stackoverflow.com/

[9] https://docs.python.org/3/library/re.html

[10] https://www.hepsiburada.com/

[11] https://pyinstaller.org/en/stable/index.html

CODE OF THE PROJECT

```python
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from webdriver_manager.chrome import ChromeDriverManager
import time
from bs4 import BeautifulSoup
import pandas as pd
import random
import requests
import tkinter as tk
from tkinter import messagebox
from PIL import Image, ImageTk


def scrapping():
    url=entry.get()
    if "hepsiburada" not in url:
     messagebox.showerror("Geçersiz Sayfa", "Bu sayfa Hepsiburada'ya ait değil.")
     return

    driver=webdriver.Chrome(service=Service(ChromeDriverManager().install()))
    driver.get(url)
    time.sleep(2)

    try:
        buton=driver.find_element(By.CLASS_NAME,"M6iJLUpgHKlEPzGcOggE")
        buton.click()
        time.sleep(2)
    except:
         print("Diğer satıcılar butonu bulunamadı!")

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


    sellers=[]


    for item in soup.find_all("div", class_="VwUAvtsSpdiwukfc0VGp IsAfBKbg4xH3kdMRzVZO mnWNji9_P_vYbkjHXtoH"):
        try:
            product_name=soup.find("h1", attrs={"data-test-id":"title"}).text.strip()
        except:
            product_name="None"
        try:
            name=item.find("a",attrs={"data-test-id":"merchant-name"}).text.strip()
        except:
          name="None"
        try:
            price=item.find("div", attrs={"data-test-id":"price-current-price"}).text.strip()
        except:
            price="None"
        try:
          cargo=item.find("div",attrs={"data-test-id":"shipment-text"}).text.strip()
        except:
            cargo="None"
        try:
            rate=item.find("span",attrs={"data-test-id":"merchant-rating"}).text.strip()
        except:
            rate="None"

        sellers.append({
        "Satıcı":name,
        "Fiyat":price,
        "Satıcı Puanı":rate,
        "Kargo Bilgisi":cargo,
        "Ürün Adı":product_name
        })

    if(len(sellers)<5):
        driver.quit()
        messagebox.showerror(title="Hata!",message="Seçtiğiniz üründe en az 5 satıcı bulunmalıdır.")


    df=pd.DataFrame(sellers)
    df.to_csv("satıcılar.csv",index=False, encoding="utf-8-sig")
    msg=f"Ürün Adı: {product_name}\n\n"

    cheapest = df.sort_values("Fiyat").head(5)

    for _, row in cheapest.iterrows():
        msg+=f"- Satıcı: {row['Satıcı']}\n"
        msg+=f"  Fiyat: {row['Fiyat']}\n"
        msg+=f"  Puan: {row['Satıcı Puanı']}\n"
        msg+=f"  Kargo: {row['Kargo Bilgisi']}\n\n"


    msg2=f"Ürün Adı: {product_name}\n\n"
    expensive=df.sort_values("Fiyat", ascending=False).head(5)

    for _, row in expensive.iterrows():
        msg2+=f"- Satıcı: {row['Satıcı']}\n"
        msg2+=f"  Fiyat: {row['Fiyat']}\n"
        msg2+=f"  Puan: {row['Satıcı Puanı']}\n"
        msg2+=f"  Kargo: {row['Kargo Bilgisi']}\n"

    driver.quit()
    messagebox.showinfo(title="En Ucuz 5 Satıcı:",message=msg)
    messagebox.showinfo(title="En Pahalı 5 Satıcı:",message=msg2)



form=tk.Tk()
form.title("Hepsiburada Fiyat Karşılaştırma Aracı")
form.geometry("500x400")
form.resizable(False,False)
buton=tk.Button(text="Url'yi Ekle",width=10,height=2,command=scrapping,bg="#ff6305")
buton.place(x=150,y=190)
entry=tk.Entry()
entry.place(x=175,y=140)
exit_button=tk.Button(text="Çıkış",width=10,height=2,command=exit,bg="#ff6305")
exit_button.place(x=250,y=190)
image = Image.open("logo.png")
image = image.resize((125, 50))
photo = ImageTk.PhotoImage(image)
label = tk.Label(form, image=photo)
label.image = photo
label.place(x=174, y=70)


form.mainloop()
```