Automating Market Research on Laptops from Daraz Store

Python Developer Internship

Summer 2023

Document:	**Project Document**	**Date:**	**08/07/2023**
Prepared by:	**Jawad Ahmed**	**Prepared for:**	**Sir Seemab Yamin**

Executive Summary:

- This report outlines the project to automate market research on laptops from the popular e-commerce platform Daraz. The objective is to develop a Selenium bot that scrapes data from the laptop category on Daraz, stores it in a database, and schedules the automation process. The report highlights the functional requirements, technology stack, implementation workflow, and project conclusion.

1. Introduction

1.1 Background:

- In today's highly competitive e-commerce market, it is crucial for businesses to gather accurate market research data to make informed decisions. Our client, a seller of the popular e-commerce platform Daraz, aims to automate the process of collecting market research data on laptops available in the store. By automating the data scraping process, our client can save time and effort while ensuring regular updates on the latest laptop offerings.

1.2 Objective:

- The objective of this project is to develop a Selenium bot that navigates the Daraz website, extracts relevant information about laptops from the laptop category, and stores it in a database for further analysis. The automation will be scheduled to run every Tuesday to provide up-to-date market research data.

2. Functional Requirements

2.1 Selenium Bot:

- Navigating the Daraz website
- Scraping laptop data: product details, prices, ratings, reviews

2.2 Database Integration

- Storing scraped data efficiently
- Supporting relevant fields: product name, price, ratings, reviews

2.3 Scheduling

- Configuring the automation process to run every Tuesday.

- Handling authentication steps seamlessly

2.4 Logging Integration

- Recording execution logs
- Tracking errors, exceptions, and successful cases

3. Technology Stack

3.1 Flask
3.2 MVC Architecture
3.3 ORM (Object-Relational Mapping)
3.4 XAMPP Server
3.5 AJAX
3.6 HTML
3.7 CSS
3.8 Selenium
3.9 Database System (MySQL)

4. Implementation Workflow

4.1 Flask Application Setup:

- Install Flask and set up a Flask application.
- Configure the necessary routes and endpoints for data retrieval and rendering.

4.2 Selenium Bot Development

- Utilize Selenium WebDriver to automate web interactions.
- Implement functions to navigate the Daraz website, locate and extract laptop data.
- Handle authentication steps if required.

4.3 Integration with Flask and Database

- Establish a connection to the chosen database system (MySQL) using an ORM (Object-Relational Mapping) framework like SQLAlchemy.
- Create a database schema to store the laptop data.
- Implement functions to save the scraped data into the database.

4.4 Scheduling Configuration

- Configure a scheduling mechanism (Python libraries) to trigger the automation process every Tuesday or we can take day as input to schedule the automation.
- Handle any necessary authentication steps during scheduled execution.
- Initiate the Selenium bot to scrape laptop data and store it in the database.

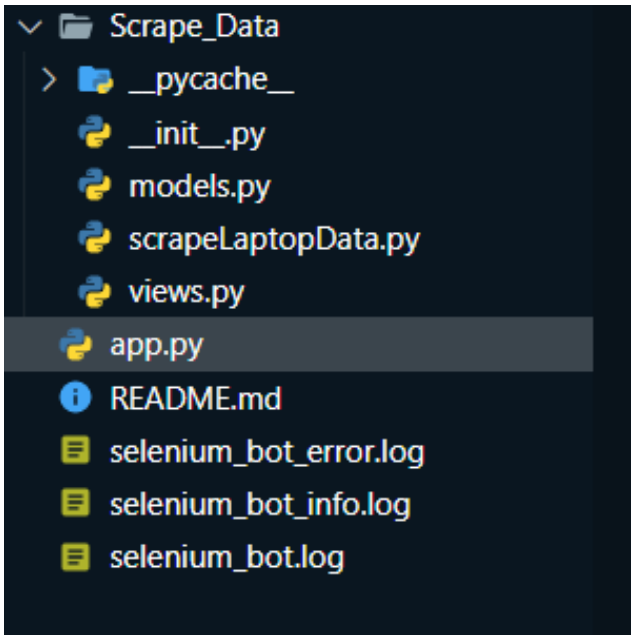
4.5 Logging Implementation

- Establish a connection to the chosen database system (MySQL) using an ORM (Object-Relational Mapping) framework like SQLAlchemy.
- Create a database schema to store the laptop data.
- Implement functions to save the scraped data into the database.

5. Result/Working of Application:

I divided the project into 2 portions, one is server side and the other one frontend.

- The server side include the following directory structure:



- There is models which includes the following table:
  - The table name is Laptop Data which contains laptop name, laptop price, laptop rating and laptop link.

```python
from flask_marshmallow import Marshmallow
from flask_sqlalchemy import SQLAlchemy

from Scrape_Data import app

db = SQLAlchemy(app)
ma = Marshmallow(app)

You, 5 hours ago | 1 author (You)
class Laptop_Data(db.Model):
    __tablename__ = 'laptop_data'
    id = db.Column(db.Integer, primary_key=True)
    laptopName = db.Column(db.Text())
    laptopPrice = db.Column(db.Text())
    laptopRating = db.Column(db.Text())
    laptopLink = db.Column(db.Text())

    def __init__(self, laptopName, laptopPrice, laptopRating,laptopLink):
        self.laptopName = laptopName
        self.laptopPrice = laptopPrice
        self.laptopRating = laptopRating
        self.laptopLink = laptopLink        You, 5 hours ago • final commit …


You, 5 hours ago | 1 author (You)
class Laptop_DataSchema(ma.Schema):
    You, 5 hours ago | 1 author (You)
    class Meta:
        fields = ('id', 'laptopName','laptopPrice','laptopRating','laptopLink')
```

- There is a __init__.py file which indicates that the directory is python module. It contained initialization code for the package, or it can be an empty file.
- Below screenshot show the details implementation of the python module which contains the connection of database, the db create model is called to create the table in the database.

```python
import os
from flask import Flask, render_template
from flask_cors import CORS
from datetime import timedelta, datetime
from sqlalchemy.exc import SQLAlchemyError

app = Flask(__name__)
app.secret_key = "codeaza-project"
CORS(app)
basedir = os.path.abspath(os.path.dirname(__file__))
app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql+pymysql://root:@localhost/codeaza_db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
app.config['PERMANENT_SESSION_LIFETIME'] = timedelta(minutes=60)

from Scrape_Data import models

try:
    # Create the database tables      You, 4 days ago • update the views for adding r
    with app.app_context():
        models.db.create_all()
except SQLAlchemyError as e:
    print(f"Failed to create database tables: {e}")
    # You can handle the exception according to your needs

# Import views after creating the tables
from Scrape_Data import views
```

- There is scrapeLaptopData.py file which contains the selenium bot code.
    - It also contains the log file configuration.
    - It also contains the schedule of selenium bot.
    - There are 2 main functions: one scrape laptop data and the other one is the schedule bot which will later be called in the views.py file to apply the given functionality.
    - Below is the screenshot of the scrapeLaptopData.py file

```python
import time        You, 22 hours ago • modify the previous version …
import logging
import warnings
import schedule
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options
from selenium.common.exceptions import NoSuchElementException
from selenium.webdriver.common.action_chains import ActionChains
from .models import Laptop_Data, Laptop_DataSchema, db
from flask import jsonify
warnings.filterwarnings("ignore")


laptop_schema = Laptop_DataSchema()
laptop_schema = Laptop_DataSchema(many=True)

# Configure logging

# Create a logger object
logger = logging.getLogger('selenium_bot')
logger.setLevel(logging.INFO)

# Create a file handler to write info logs to the file
info_handler = logging.FileHandler('selenium_bot_info.log')
info_handler.setLevel(logging.INFO)

# Create a file handler to write error logs to the file
error_handler = logging.FileHandler('selenium_bot_error.log')
error_handler.setLevel(logging.ERROR)

# Create a formatter to specify the log message format
formatter = logging.Formatter('%(asctime)s - %(levelname)s - %(message)s')

# Set the formatter for the handlers
info_handler.setFormatter(formatter)
error_handler.setFormatter(formatter)

# Add the handlers to the logger
logger.addHandler(info_handler)
logger.addHandler(error_handler)

# Example usage
try:
    # Some code that may raise an error
    raise ValueError('An error occurred')
except Exception as e:
    logger.error(f'Error occurred: {str(e)}')
```

```python
# Set up Selenium Chrome options

chrome_options = Options()
chrome_options.add_argument('--headless')  # Run in headless mode to avoid opening a browser window
chrome_options.add_argument('--log-level=3')  # Suppress logging messages

# Create a new instance of the Chrome driver
driver = webdriver.Chrome(options=chrome_options)

def scrape_laptop_data(pages):

    print('Laptop Data Scraping Started')
    try:
        # Open Daraz website
        driver.get('https://www.daraz.pk/')

        # Accept cookies if prompted
        # Add a delay to allow time for the cookie banner to appear
        time.sleep(2)
        try:
            driver.find_element(
                By.XPATH, "//button[text()='Accept All']").click()
        except NoSuchElementException:
            pass

        # Navigate to the electronic devices category
        driver.find_element(
            By.XPATH, '//*[@id="Level_1_Category_No1"]/a/span').click()

        # Navigate to the laptop category
        driver.find_element(
            By.XPATH, '//*[@id="J_5022174600"]/div/ul/ul[7]/li[5]/a/span').click()

        # Scrape laptop data from first 10 pages
        data = []
        for page in range(1, pages + 1):
            # Scroll to the bottom of the page to load all laptops
            last_height = driver.execute_script(
                "return document.body.scrollHeight")
            while True:
                driver.execute_script(
                    "window.scrollTo(0, document.body.scrollHeight);")
                time.sleep(2)  # Add a delay to allow time for the page to load
                new_height = driver.execute_script(
                    "return document.body.scrollHeight")
                if new_height == last_height:
                    break
                last_height = new_height
```

```python
# Scrape laptop data
laptops = driver.find_elements(By.XPATH, '//div[@class="box--ujueT"]/div[@class="gridItem--Yd0sa"]')

for laptop in laptops:
    # Extract the required laptop details
    info_div = laptop.find_element(By.XPATH, './/div[@class="info--ifj7U"]')
    name_element = info_div.find_element(By.XPATH, './/div[@class="title--wFj93"]/a')
    name = name_element.text
    link = name_element.get_attribute("href")
    price_element = info_div.find_element(By.XPATH, './/div[@class="price--NVB62"]/span')
    price = price_element.text

    try:
        # Adjust rating out of 10
        rating_element = info_div.find_element(By.XPATH, './/div[contains(@class, "rating--ZI3Ol")]')
        rating_icons = rating_element.find_elements(By.XPATH, ".//i[contains(@class, 'star-icon--k88DV')]")
        rating = 0
        total_ratings = len(rating_icons)

        for icon in rating_icons:
            class_name = icon.get_attribute("class")
            if "star-10" in class_name:
                rating += 10
            elif "star-9" in class_name:
                rating += 9
            elif "star-8" in class_name:
                rating += 8
            elif "star-7" in class_name:
                rating += 7
            elif "star-6" in class_name:
                rating += 6
            elif "star-5" in class_name:
                rating += 5
            elif "star-4" in class_name:
                rating += 4
            elif "star-3" in class_name:
                rating += 3
            elif "star-2" in class_name:
                rating += 2
            elif "star-1" in class_name:
                rating += 1

        if total_ratings > 0:
            rating /= total_ratings
        else:
            rating = 0
    except NoSuchElementException:
        rating = 0
```

```python
                # Store the laptop details into the database using Flask ORM
                existing_laptop = Laptop_Data.query.filter_by(laptopName=name).first()
                if existing_laptop is None:
                    # Store the laptop details in a new record
                    laptop_data = Laptop_Data(laptopName=name, laptopPrice=price, laptopRating=rating, laptopLink=link)
                    db.session.add(laptop_data)
                    db.session.commit()

                    # Serialize the laptop data using the schema
                    serialized_data = laptop_schema.dump([laptop_data])
                    json_data = jsonify(serialized_data)
                    # Perform any further processing or response handling with the JSON data


                else:
                    # Duplicate laptop entry found, skip adding to the database
                    logging.info(f"Skipping duplicate laptop entry: {name}")

                # Log the laptop details
                logging.info(f"Laptop Name: {name}, Price: {price}, Rating: {rating}")

            # Go to the next page
            next_page = driver.find_element(By.XPATH, '//li[@title="Next Page"]/a[@class="ant-pagination-item-link"]')
            actions = ActionChains(driver)
            actions.move_to_element(next_page).click().perform()


    except Exception as e:
        logging.error(f"An error occurred: {str(e)}")

    finally:
        # Close the browser
        driver.quit()

# Schedule the bot to run every n days
def schedule_bot(day):
    schedule.every(day).days.do(scrape_laptop_data)
    while True:
        schedule.run_pending()
        time.sleep(1)
```

- Now talking about the views.py file which contains the overall logic of the file means there I create an api which will be called in the frontend by Ajax request in order to get response back.
- Below is the screenshot of the views.py file code

```python
from flask import  jsonify, request      You, 7 hours ago • optimized the code …
from .models import  Laptop_Data, Laptop_DataSchema, db
from .scrapeLaptopData import scrape_laptop_data, schedule_bot
from Scrape_Data import app


laptop_schema = Laptop_DataSchema()
laptop_schema = Laptop_DataSchema(many=True)


@app.route('/')
def index():
    return 'Server Is Active. Access its Services Through Client or Frontend'


@app.route('/home/api')
def home():
    # Retrieve the laptop data from the database
    laptop_data = Laptop_Data.query.all()

    # Serialize the laptop data using the schema
    serialized_data = laptop_schema.dump(laptop_data)

    # Return the JSON data to the home.html template
    return jsonify(serialized_data)


@app.route('/get-total-laptop-data/api', methods=['GET'])
def get_total_laptop_data():
    # Calculate the total number of laptop data
    total_laptop_data = Laptop_Data.query.count()

    # Create a JSON response
    response = {
        'total': total_laptop_data
    }

    # Return the response
    return jsonify(response)
```

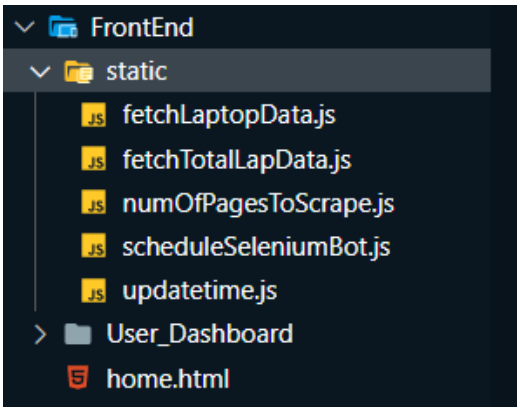
```python
@app.route('/startsraping/api', methods=['POST'])
def startscraping():
    num_pages = int(request.form['numPages'])

    # Call the scrapeLaptopData function to start the scraping process
    print(f"Request Made To Start Scraping with num_pages: {num_pages}")
    scrape_laptop_data(num_pages)  # Enter number of pages to scrape

    # Return a response indicating the scraping process has started
    return "Scraping process started"



@app.route('/schedulebot/api', methods=['POST'])
def botscraping():
    num_days = int(request.form['numDays'])

    # Call the scrapeLaptopData function to start the scraping process
    print(f"Request Made To Start Selenium Bot Scraping after Num_Days: {num_days}")
    schedule_bot(num_days)  # Enter number of days to scrape

    # Return a response indicating the scraping process has started
    return "Automatically Selenium Bot Scraping process started"
```

- Now we are left with the app.py file which contains the application running code with Debug mode.
- Below is the screenshot of the app.py file

```python
# Main file to run the server

from Scrape_Data import app

if __name__ == '__main__':
    app.run(debug=True)
```

Now when I run the application, I got the server ready to response.
It's time to investigate the Frontend Side.

The front-end side contains the following files.



The static files contain all the javascript files that need to be communicated with the server.
Let's look at those files in detail.

fetchLaptopData.js file is used to get the scraped laptop data from the database. Below is the screenshot of the file code.

The data in the database shows like below:

←T→				id	laptopName	laptopPrice	laptopRating	laptopLink
☐	Edit	Copy	Delete	1	Dell Chromebook 11 3189 - 360 TouchScreen - Intel ...	Rs. 15,299	9.8	https://www.daraz.pk/products/dell-chromebook-11-3...
☐	Edit	Copy	Delete	2	Acer C720 ChromeBook Laptop - 4gb / 16gb - Long Ba...	Rs. 15,500	9	https://www.daraz.pk/products/c720-16gb4gb-i146980...
☐	Edit	Copy	Delete	3	Dell 3189 Convertible Chromebook 11.6 inches HD IP...	Rs. 15,299	9.6	https://www.daraz.pk/products/dell-3189-convertibl...
☐	Edit	Copy	Delete	4	CTL ED20PA2 LAPTOP | 11.6" TOUCH SCREEN | 4GB RAM ...	Rs. 15,499	0	https://www.daraz.pk/products/ctl-ed20pa2-laptop-1...
☐	Edit	Copy	Delete	5	MacBook Pro Mid 2009 | 13 inches | Intel Core 2 Du...	Rs. 31,500	0	https://www.daraz.pk/products/macbook-pro-mid-2009...

The above data is used to display on the website using the views.py file on the server side and the below code on the frontend side.

```javascript
$(document).ready(function () {
    // Function to fetch the laptop data and populate the table
    function fetchLaptopData() {
        $.ajax({
            url: "http://127.0.0.1:5000/home/api",
            type: "GET",
            dataType: "json",
            success: function (data) {
                // Process the JSON data and populate the table
                populateTable(data);
            },
            error: function (xhr, status, error) {
                console.log("Error retrieving laptop data:", error);
                $('#errorAlert1').text('Error retrieving laptop data').show();
            }
        });
    }

    // Function to populate the table with laptop data
    function populateTable(laptopData) {
        var tableBody = $("#scraped_table tbody");
        tableBody.empty();

        // Iterate over the laptop data and populate the table rows
        for (var i = 0; i < laptopData.length; i++) {
            var laptop = laptopData[i];

            // Extract the truncated laptop name
            var truncatedName = laptop.laptopName.split(/,|-|\|/)[0].trim();

            var rowHtml = "<tr>" +
                "<td>" + (i + 1) + "</td>" +
                "<td>" + truncatedName + "</td>" +
                "<td>" + laptop.laptopPrice + "</td>" +
                "<td>" + laptop.laptopRating + "</td>" +
                "<td><a href='" + laptop.laptopLink + "' target='_blank'>View</a></td>" +
                "</tr>";
            tableBody.append(rowHtml);
        }
    }

    You, 5 hours ago • final commit …
    // Call fetchLaptopData function on page load
    fetchLaptopData();

});
```

The above code shows below the result on the website.

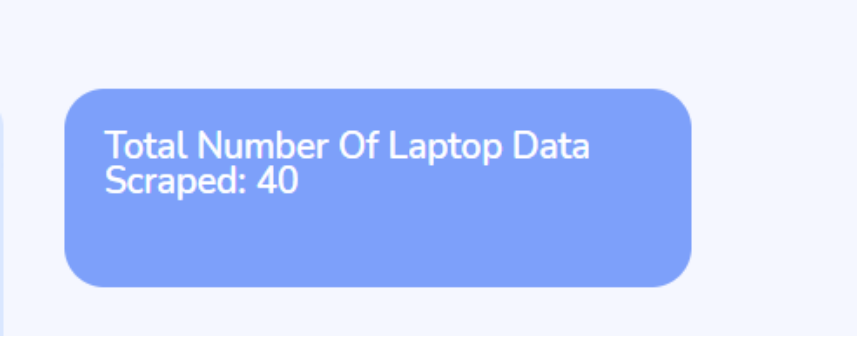
Scraped Daraz Laptop Data Upto N Pages

#	Laptop Name	Laptop Price	Laptop Ranking	Laptop Link
1	Dell Chromebook 11 3189	Rs. 15,299	9.8	View
2	Acer C720 ChromeBook Laptop	Rs. 15,500	9	View
3	Dell 3189 Convertible Chromebook 11.6 inches HD IPS Touchscreen	Rs. 15,299	9.6	View
4	CTL ED20PA2 LAPTOP	Rs. 15,499	0	View
5	MacBook Pro Mid 2009	Rs. 31,500	0	View
6	HP Chromebook 11 G7 EE 11.6 Inch Laptop	Rs. 14,699	9.8	View
7	Dell	Rs. 22,999	10	View
8	Dell Inspiron Chromebook 11 3100	Rs. 16,000	9.6	View

- Let see about the fetchTotalLaptopData.js file, it gets the count of the total data scraped, initially I just scraped the 1 page of the daraz which contains 40 laptop data, it can be increased later on by giving input to system.
- Below is the code of this file
-

```
You, 6 hours ago | 1 author (You)
$(document).ready(function () {

    // Function to fetch the total number of laptop data
    function fetchTotalLaptopData() {
        $.ajax({
            url: "http://127.0.0.1:5000/get-total-laptop-data/api",
            type: "GET",
            dataType: "json",
            success: function (data) {
                var totalLaptopData = data.total;
                $("#total_laptop_data").text(totalLaptopData);
            },
            error: function (xhr, status, error) {
                console.log("Error retrieving total laptop data:", error);
            }
        });
    }

    // Call fetchTotalLaptopData function on page load
    fetchTotalLaptopData();        You, 6 hours ago • update home.html …
});
```

-

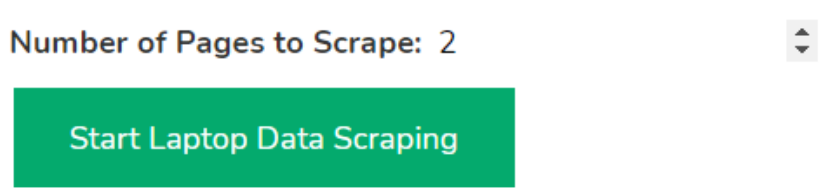
- It shows below the result on the website.
- 

- Now coming towards the numOfPagesToScrape.js file, it take pages number as input and send it to the server there it will scrape upto the given pages let say we can scrape it upto 10 pages.

- It contains below code

- ```javascript
$(document).ready(function () {
    // Function to start the scraping process
    function startScraping() {
        var numPages = parseInt($('#pages-input').val());
        if (isNaN(numPages) || numPages < 1) {
            alert('Invalid number of pages. Please enter a positive integer.');
            return;
        }

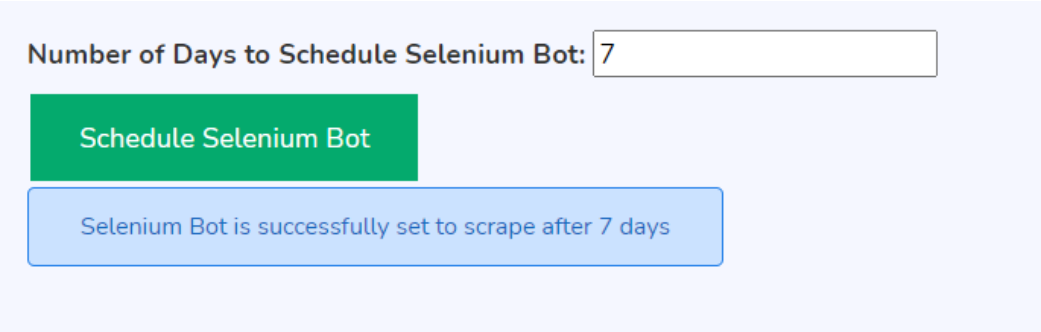
        $('#scraped_table tbody').empty();
        $('#successAlert1').hide();
        $('#errorAlert1').hide();
        $('#waitAlert1').text('Wait Product Is Scraping').show();

        $.ajax({
            url: "http://127.0.0.1:5000/startsraping/api",
            type: "POST",
            data: { numPages: numPages },
            success: function (response) {
                console.log(response);
                // After starting the scraping, fetch and display the laptop data
                fetchLaptopData();
                $('#waitAlert1').text('Wait Product Is Scraping').hide();
                $('#successAlert1').text('Laptop Data Scraped Successfully').show();
            },
            error: function (xhr, status, error) {
                console.log("Error starting the scraping process:", error);
                $('#errorAlert1').text('Error starting the scraping process').show();
                $('#waitAlert1').text('Wait Product Is Scraping').hide();
                $('#successAlert1').text('Laptop Data Scraped Successfully').hide();
            }
        });
    }
    // Button click event handler
    $("#start-scraping-btn").click(function (event) {
        event.preventDefault();
        startScraping();
    });
```
- It result in the following way on the website
- 
```

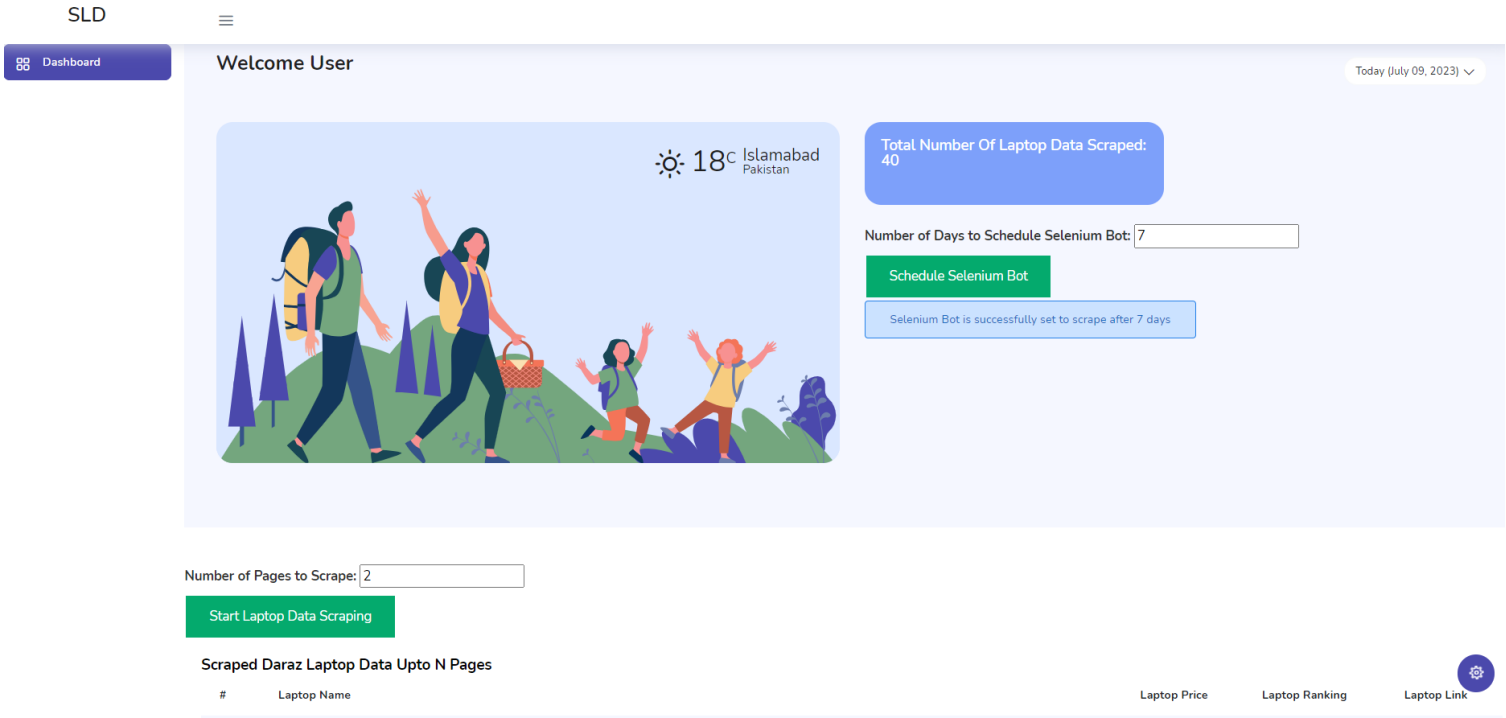
- The last file is scheduleSelenium.js which is used to take day as input and send it to server in order to set the bot to automatically run the selenium script we can also keep it fix upto Tuesday but here I provide flexibility by getting number of days as input. Below is the code of it:

```javascript
//handle the selenium Bot
$(document).ready(function () {
    // Handle form submission
    $("#scraping-form").submit(function (event) {
        event.preventDefault();
        scheduleBot();
    });

    // Function to schedule the Selenium bot scraping
    function scheduleBot() {
        var numDays = parseInt($("#days-input").val());

        $('#successAlert2').hide();
        $('#errorAlert2').hide();
        $('#waitAlert2').text(`Selenium Bot is successfully set to scrape after ${numDays} days`).show();

        $.ajax({
            url: "http://127.0.0.1:5000/schedulebot/api",
            type: "POST",
            data: { numDays: numDays },
            success: function (response) {
                console.log(response);
                $('#waitAlert2').hide();
                $('#successAlert2').text('Selenium Bot Scraping Scheduled Successfully').show();
            },
            error: function (xhr, status, error) {
                console.log("Error scheduling the Selenium bot scraping:", error);
                $('#errorAlert2').text('Error scheduling the Selenium bot scraping').show();
                $('#waitAlert2').hide();
            }
        });
    }
});      You, 6 hours ago • finalize the home page …
```

- It gives following output:



- The overall application interface is shown like this:



Github Link:

The project code is uploaded over the GitHub link is: https://github.com/jawadahmed2/Automating-Market-Research-on-Laptops-from-Daraz-Store

Conclusion:

- Automating market research on laptops from Daraz will provide valuable data for our client.
- Automation saves time and effort, allowing the client to make informed decisions.
- The solution enables the client to optimize their laptop-selling strategy on Daraz.