SCRAPING OF E-COMMERCE WEBSITES USING PUPPETEER AND NODE JS

Term Project – CECS 590 Selected Topics in Computer Science

By:

Praveen Patare - 018773155
Rushabh Jaiswal – 018757815

Guidance by:

Dr. Roman Tankelevich.

ABSTRACT

Web scraping is a technique of extracting large amount of data from websites and using the data for analysis and comparison. The main goal of this project was to implement a web-based application for listing the details of the product searched by the user such as its price, name and price fetched from different ecommerce websites. Today there are many ecommerce websites selling large number of products that we may usually find at a neighbor-hood store or a supermarket. The main benefit of shopping on websites is that we don't have to get outside our house for buying anything. We can just buy the product we want with a click of a button and the product gets delivered to our doorstep within a day or week. However, as the number of ecommerce giants grow it is making us more and more difficult to buy a product from any website owing to the competitive offers offered by these websites. The implemented web application relieves the user from this problem. It involves scraping these websites for the searched product and providing a list of the product on all the ecommerce websites thereby reducing the overhead of user to visit every website and checking for the offer on the product. This helps the user to compare the prices, reviews and availability of the searched product on different, well-known ecommerce websites and select the best deal available for the product. The web application involves implementing a scrapper that gets product related information matching the user search string by visiting each link related to the product synchronously. All the required information is searched on the product web page and collected by the scrapper. The information is then stored in the database and rendered to the user as a web page showing the results of the searched product including its name, image, link for the product on the ecommerce website and price of the product to make the comparison.

BACKGROUND

The use of e-commerce websites has immensely increased from past two or three years due to the increased use of internet and other related technologies. E-commerce refers to the online stores that sells large amount of category of products nationwide. Due to the increase of these online stores large amount of data is generating each day which remains unanalyzed. This large amount of data can be useful for the behavioral as well as market research. Analyzing such a large amount of data is very important to analyze the changes in prices, launch of new product in the market, rising trends and comparing same product from the different e-commerce websites.

In today's world, providing convenience to the users or customers is the primary goal. Most of the time, these online stores provide the offers for the limited amount of time and users can't get benefit of that because it's very difficult for them to keep track of each and everything available on the different websites. The range of offers for a particular product offered by these ecommerce websites makes it difficult for the users to select the best choice as the user has to browse through all these sites to check for the best deal. This led us to the idea of developing a website that provided a list of the offers on a particular product offered by all the available ecommerce websites thus reducing the overhead of the user.

INTRODUCTION

Web scraping refers to the extraction of required data from the internet whereby the data is extracted from the internet and stored it into the database to analyze it in various ways.
The working of web scraping is depicted in the picture below:

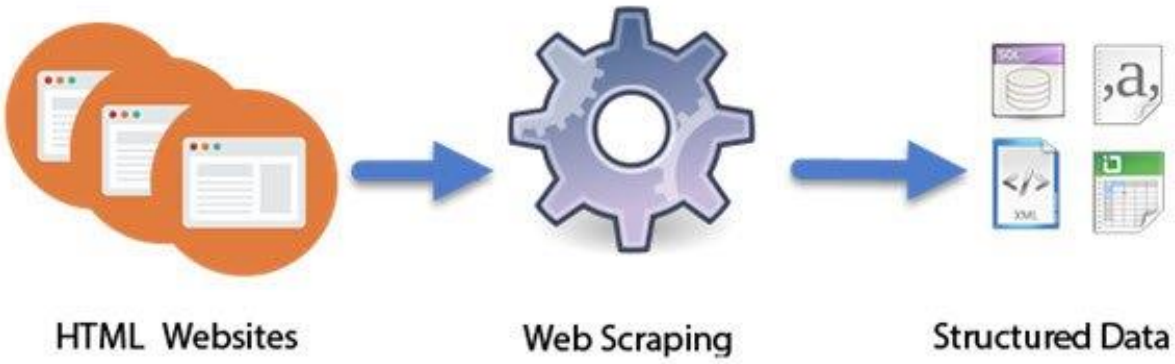


Figure 1 Web Scraping Architecture

Web scraping could be done either by visiting that particular website, downloading the source code of the webpage and then analyzing the source code and extracting the useful information embedded in the html tags. The other way that we have used in our project is to use libraries that initiate a headless browser. This browser visits the web page to be scrapped. The library provides a handle to the webpage that is being browsed. Using this handle, we can issue various commands to the

web page such as clicking on a link, entering data into the input fields, taking page screenshot etc. Once we are on the page that contains the required data, we use CSS selectors to get the data and store it in the required format. In our case, the library that we have used is called puppeteer.

PUPPETEER

Puppeteer is a node-based library that provides us with all the browser functions like go to a webpage by providing the page URL, click on a link on the web page, take a screenshot of the web pages as results etc. The library is mainly used for browser automation. It works by initiating a chromium browser that visits the web page to be scrapped and analyses the data on the web page using various functions provided by the library. Below is the list of some of the functions we used in our project-

1. launch() - Used to launch a new browser session. The browser could be headless or with header.
2. goto() - It is used to go to the web page whose URL is provided as an argument in the function.
3. page() - Used to open a new tab on the browser or a new instance of the page.
4. click() - Used to click on a particular element or link or button of the webpage.
5. type() - Used to type text in an input html element like text box, text area.
6. close() - Used to close the current browser after the scraping is performed.
7. evaluate() - This function allows us to use JavaScript that is to be used to scrap the webpage. All the logic of extracting the data from the web page goes in to this method.

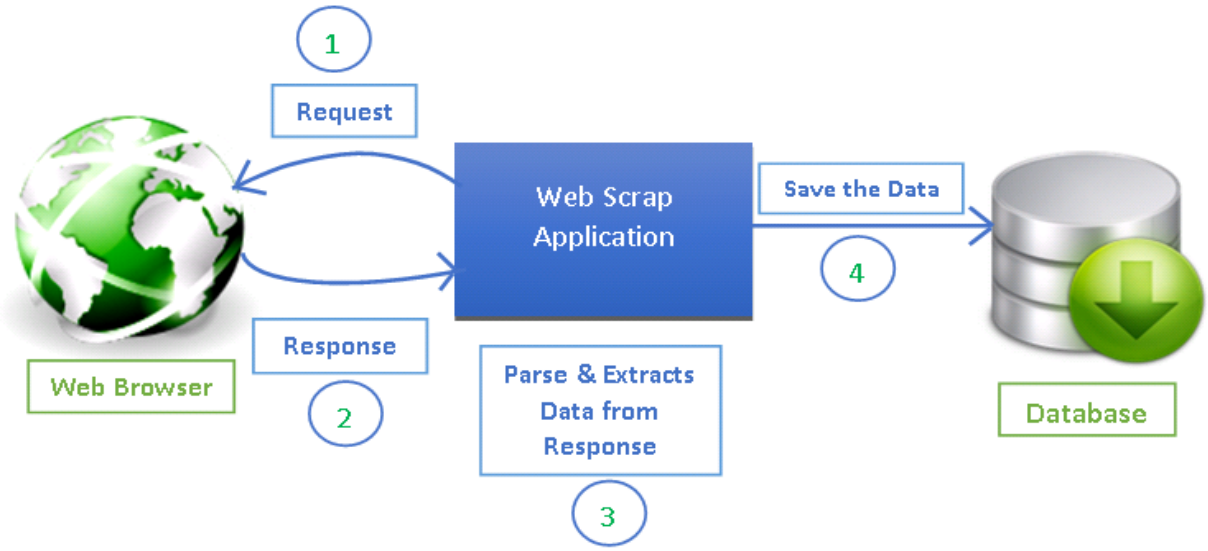


Figure 2 Working of Web Scraping

The detailed steps followed by scrapper are listed below -

1. Initially we create a browser object using the puppeteer function called launch. The function provides an option of either launching a headless browser or with a header.
2. Once the browser is launched, we ask the browser to open a new page and visit the URL using the goto function of the library.
3. Once the homepage of the website is rendered by the browser, we use the type command to type the product name entered by the user in the search box. The search box of the website is located by using the CSS selectors of the page.
4. Once we get a list of the links related to the product, we save the links in an array and visit each relevant link by using the same functions as discussed above.
5. For each page visit, the required data related to the product like title, price, product link etc. is stored in an object.
6. The final list of all the objects containing product data is stored in a NoSQL database.

MONGO DB

MongoDB is a NoSQL based document-oriented database. The database provides many important functions like indexing, replication, load balancing, aggregation, server-side JavaScript execution. The main reason for using MongoDB as our underlying database was because of its querying in JavaScript language. MongoDB stores unstructured data in the form of JSON documents which can be easily queried and updated using JavaScript functions. Mongo DB also provides a JavaScript based library mongoose which makes it easier to query large amount of data. MongoDB supports many types of indexing for improved search results. The one type of indexing that we used was text indexing. The data stored in our mongo DB database was also of string format and so the text-based indexing was the best way to search for the product in the database by using the keywords from the search string taken from the user. In addition to the above reasons, mongo dB also provides some additional features that make it the best choice over SQL databases. The support for unstructured data, the fast searching of the data from the database and simple querying using JavaScript makes it the based choice to use for web apps implemented in the JavaScript programming language.

In addition to using puppeteer library for scraping and MongoDB for storing the scraped data, we also implemented a JavaScript based web application that helps us to take the request from the user, route the request to the server and initiate the scrapper, store the scraped data in a MongoDB and finally display the results on a web page to the user. Below is the list of the additional frameworks and technologies used for the web application.

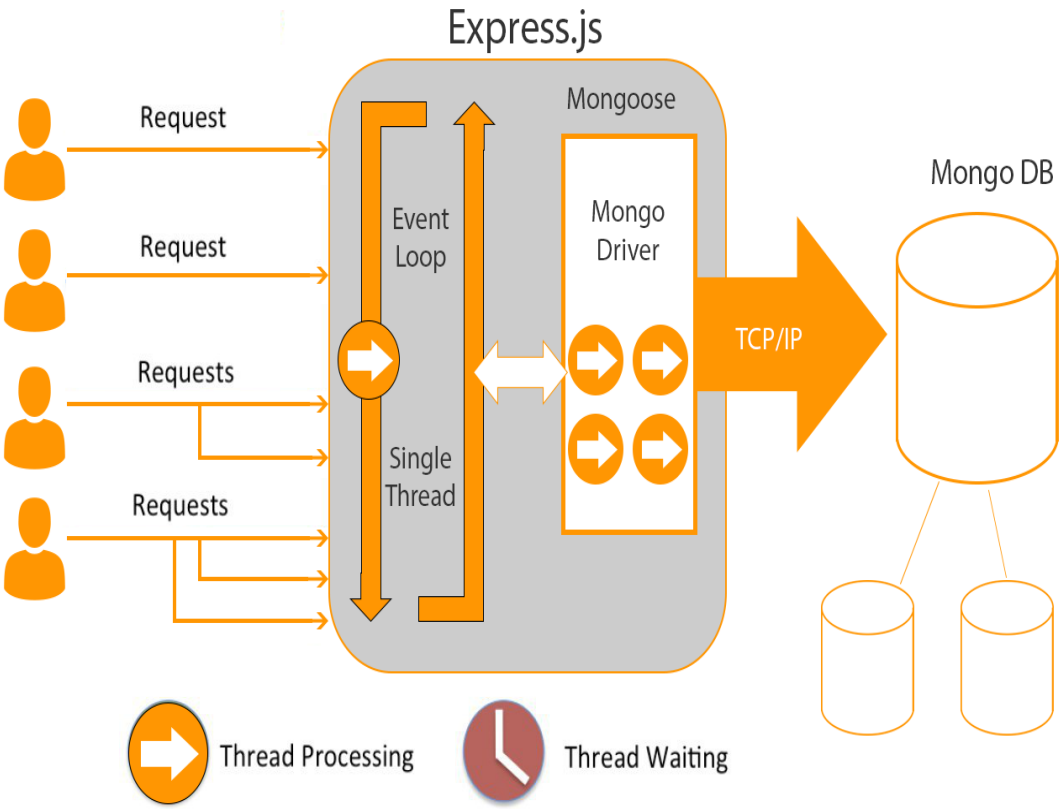


Figure 3 Integration of Express with MongoDB

TECHNOLOGIES USED

Node.js

Node.js is a JavaScript environment that allows the JavaScript to run without a browser which is built on google chrome's JavaScript engine for building fast and scalable applications.
Features:

- **Asynchronous and Event Driven**: All the API's of Node.js are asynchronous meaning non-blocking. Any node.js based web server doesn't wait for the API to return data. Instead, it moves to another API and the notification mechanism of node.js helps its server to get the response data from the previously called API.
- **No Buffering:** Node.js applications never buffer the entire data. Instead it outputs the data in smaller chunks of data.
- **Single threaded but Highly scalable**: Node.js creates a single threaded program that can provide services to the large number of requests. Whereas traditional services as not as efficient as this server. Event mechanism used in Node.js helps the server to respond to the request in non-blocking way that makes it highly scalable.

Express.js

Express.js is a Node.js framework that is widely used for building web-based application. Express.js has some of its own features in addition to the Node.js features.

Node.js uses a node package manager (NPM) which is used to install each of the dependencies required to make a web application. Express.js is one of the dependencies amongst them.
Features:

- **Routing:** routing refers to how a web server responds to the client's request. Node.js also has its own routing mechanism but routing using Express.js becomes more efficient and it is best suited for the dynamic URL's.
- **Debugging:** Express.js has its own debugging mechanism. Almost every developer encounter bugs during programming and finding out the source of the bugs can be difficult. In that case, this debugging mechanism can help the developers to find out the exact source of the bug.
- **Middleware:** Express.js supports middleware to arrange different function calls. A middleware is a code or program between a user's request and application's response. For example, in a login system, authentication process acts as a middleware.

CONCLUSION

In the end to conclude about the project we realized that the technique of web scraping requires a lot of patience and a detailed study of the web site that we want to scrape. It requires one to have a deep knowledge and understanding about the structure of the web pages including its DOM structure, the CSS styles applied to the page and the scripts. It also involves various issues that need to be overcome in order to develop a successful scrapper. The main issues that one needs to address additionally are the browser and network timeouts. The time required to browse each, and every page of the product is also an important factor to determine the speed of scrapping. The positive side of this project was that we could implement a basic scrapper that could provide us the accurate data to some extent if we set aside the problem of timeouts and network latency. Working on this project also helped us to get more insight into the puppeteer library of the node JS environment. We could learn the various functions that the library provided which are used for web application automation testing on a large scale in the software companies around the world. It helped us to understand the concepts on async/await functions of the java-script programming language. It also helped to understand and work on the basics of creating a web application using java-script frame works like node and express. Last but not the least the project was a perfect platform for us to get introduced to the current tools and libraries used in the industry and to create an industry level software that could solve some issues faced by the general population.

FUTURE WORK

The implemented web application had a few discrepancies in terms of performance which we will be looking into in the future. The main aspect of this concept is the scalability of the data scrapped by the scrapper and also the time required for the data to be scraped and rendered to the user. In order to increase the performance of the scraping the application can be deployed on AWS Lambda. AWS lambda is a service provided by the Amazon's AWS used for serverless computing. It provides resources like for user to run the code without any human intervention. AWS lambda takes care of everything required to run and scale the code for high availability. We can also setup our code to automatically trigger from other AWS services or call it directly from web or mobile platform. This in turn resolves the problem of scalability as the number of ecommerce websites increases. The future work could also involve making the scrapping more efficient so that some unwanted products that get searched could be avoided and the user could get a list of only the product he searched for.

RESULTS

```
Rushabhs-MBP:project rushabhjaiswal$ node walmart.js
[ '/Samsung-SM-G973F-Unlocked-Smartphone-International/dp/B07NXVM3DG/ref=sr_1_3?keywords=samsung+galaxy+s10&qid=1557457227&s=gateway&sr=8-3',
  '/Samsung-SM-G973F-Unlocked-Smartphone-International/dp/B07NZXXZB2/ref=sr_1_4?keywords=samsung+galaxy+s10&qid=1557457227&s=gateway&sr=8-4',
  '/Samsung-Galaxy-Factory-Unlocked-Phone/dp/B07N4M412B/ref=sr_1_5?keywords=samsung+galaxy+s10&qid=1557457227&s=gateway&sr=8-5',
  '/Samsung-SM-G975F-Smartphone-International-No-Warranty/dp/B07NZXBRPS/ref=sr_1_6?keywords=samsung+galaxy+s10&qid=1557457227&s=gateway&sr=8-6',
  '/Samsung-SM-G975F-Unlocked-Smartphone-International/dp/B07NZX5BKH/ref=sr_1_7?keywords=samsung+galaxy+s10&qid=1557457227&s=gateway&sr=8-7',
  '/Samsung-SM-G975F-Unlocked-Smartphone-International/dp/B07NZVWBJD/ref=sr_1_8?keywords=samsung+galaxy+s10&qid=1557457227&s=gateway&sr=8-8',
  '/Samsung-Galaxy-S10-SM-G9730-International/dp/B07PHFW8LV/ref=sr_1_13?keywords=samsung+galaxy+s10&qid=1557457227&s=gateway&sr=8-13',
  '/Samsung-SM-G975F-Dual-SIM-Unlocked-Smartphone/dp/B07NWR7QYP/ref=sr_1_14?keywords=samsung+galaxy+s10&qid=1557457227&s=gateway&sr=8-14',
  '/Samsung-SM-G975F-Unlocked-Smartphone-International/dp/B07QCCVSX8/ref=sr_1_15?keywords=samsung+galaxy+s10&qid=1557457227&s=gateway&sr=8-15',
  '/Samsung-i-Blason-Stylish-Protective-Protector/dp/B07MZR8KRZ/ref=sr_1_18?keywords=samsung+galaxy+s10&qid=1557457227&s=gateway&sr=8-18' ]
```

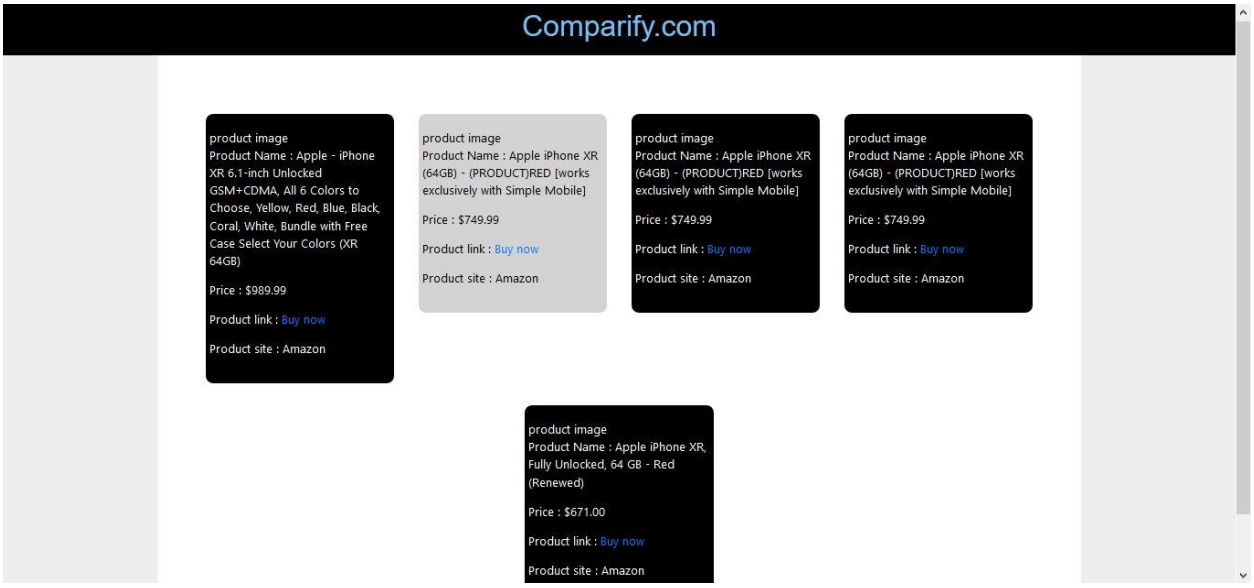
List of product links scraped for Samsung Galaxy S10 searched by user

```
^CRushabhs-MBP:project rushabhjaiswal$ node server.js
[ '/gp/slredirect/picassoRedirect.html/ref=pa_sp_atf_aps_sr_pg1_1?ie=UTF8&adId=A091457225L0S2C9HRYE&url=%2FApple-iPhone-XR-64GB-exclusively%2Fdp%2FB07K97BQDF%2Fref%3Dsr_1_1_sspa%3Fkeywords%3DApple%2Biphone%2Bxr%26qid%3D1557517781%26s%3Dgateway%26sr%3D8-1-spons%26psc%3D1&qualifier=1557517781&id=1158156455834259&widgetName=sp_atf',
  '/Apple-iPhone-XR-Fully-Unlocked/dp/B07P978C2R/ref=sr_1_3?keywords=Apple+iphone+xr&qid=1557517781&s=gateway&sr=8-3',
  '/Apple-iPhone-XR-64GB-exclusively/dp/B07K97BQDF/ref=sr_1_4?keywords=Apple+iphone+xr&qid=1557517781&s=gateway&sr=8-4' ]
```

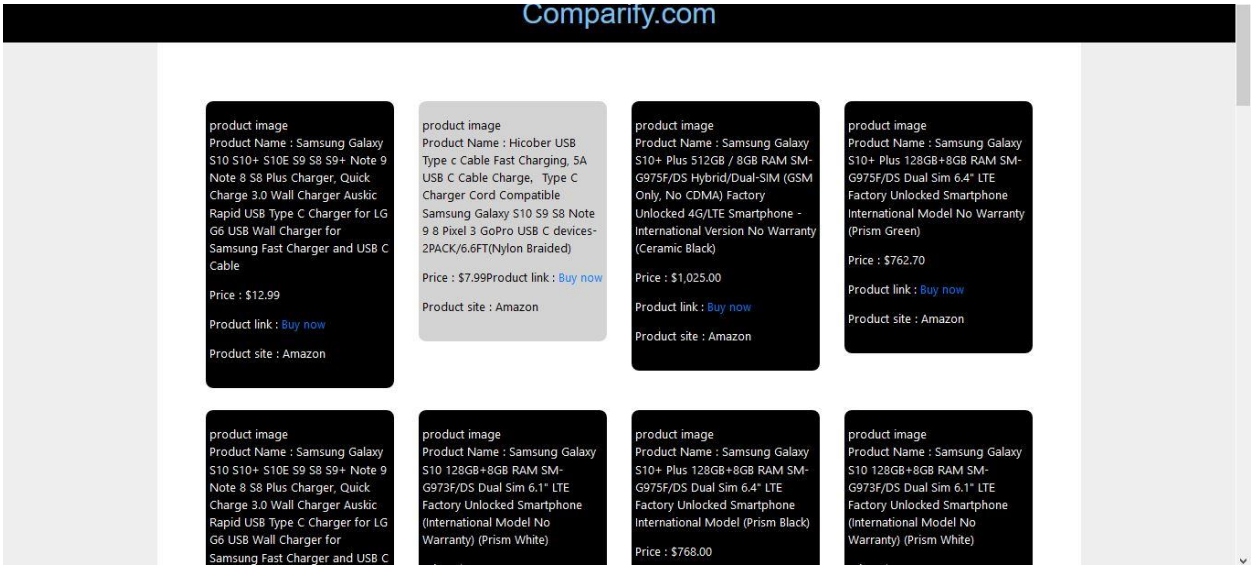
List of product links scraped for Apple iPhone XR searched by user

```
Visiting page : https://www.amazon.com/gp/slredirect/picassoRedirect.html/ref=pa_sp_atf_aps_sr_pg1_1?ie=UTF8&adId=A091457225L0S2C9HRYE&url=%2FApple-iPhone-XR-64GB-exclusively%2Fdp%2FB07K97BQDF%2Fref%3Dsr_1_1_sspa%3Fkeywords%3DApple%2Biphone%2Bxr%26qid%3D1557517781%26s%3Dgateway%26sr%3D8-1-spons%26psc%3D1&qualifier=1557517781&id=1158156455834259&widgetName=sp_atf
[ '/ip/Straight-Talk-Apple-iPhone-XR-w-64GB-Prepaid-Smartphone-Black/917049481',
  '/ip/AT-T-Apple-iPhone-XR-64GB-Black/633833645' ]
Evaluating product page
Apple iPhone XR (64GB) - (PRODUCT)RED [works exclusively with Simple Mobile]
$749.99
https://images-na.ssl-images-amazon.com/images/I/41x3E3NeRVL._SX38_SY50_CR,0,0,38,50_.jpg
https://www.amazon.com/gp/slredirect/picassoRedirect.html/ref=pa_sp_atf_aps_sr_pg1_1?ie=UTF8&adId=A091457225L0S2C9HRYE&url=%2FApple-iPhone-XR-64GB-exclusively%2Fdp%2FB07K97BQDF%2Fref%3Dsr_1_1_sspa%3Fkeywords%3DApple%2Biphone%2Bxr%26qid%3D1557517781%26s%3Dgateway%26sr%3D8-1-spons%26psc%3D1&qualifier=1557517781&id=1158156455834259&widgetName=sp_atf
Amazon
Visiting page : https://www.walmart.com/ip/Straight-Talk-Apple-iPhone-XR-w-64GB-Prepaid-Smartphone-Black/917049481
Visiting page : https://www.amazon.com/Apple-iPhone-XR-Fully-Unlocked/dp/B07P978C2R/ref=sr_1_3?keywords=Apple+iphone+xr&qid=1557517781&s=gateway&sr=8-3
Evaluating product page
Apple iPhone XR, Fully Unlocked, 64 GB - White (Renewed)
$665.00
https://images-na.ssl-images-amazon.com/images/I/219qbI5zpCL._SX38_SY50_CR,0,0,38,50_.jpg
https://www.amazon.com/Apple-iPhone-XR-Fully-Unlocked/dp/B07P978C2R/ref=sr_1_3?keywords=Apple+iphone+xr&qid=1557517781&s=gateway&sr=8-3
Amazon
```

List of data scraped for the product



User interface listing the products scraped – Apple iPhone XR



User interface listing the products scraped – Samsung Galaxy S10

APPENDIX

Scrapper code – server.js

```
'use strict';

const puppeteer = require('puppeteer');
const db = require('../db');
const Products = require('../models/products');

var basePages = {'amazon' : 'https://www.amazon.com', 'walmart' : 'https://www.walmart.com',
'ebay' : 'https://www.ebay.com'};

main();

async function main(){

  try{

     exports.index =  await function (req, res) {
       res.sendFile(path.resolve('views/index.html'));
      };

      exports.getData = await function(req, res){
       console.log('In controller' + req.body.name);
       var regex = new RegExp(req.body.name, "i");
       console.log(regex);

        Products.aggregate([{$match: { $text: { $search: req.body.name } }},{$project:
{productTitle : 1, productPrice : 1, productLink : 1, productImageLink : 1, productSite : 1, score:
{ $meta: "textScore" } }},{$match: { score: { $gt: 1 } }},{$project: {_id : 0, productTitle : 1,
productPrice : 1, productLink : 1, productImageLink : 1, productSite : 1}}], function(err,
products){
        if(err){
          console.log('Error in fetching data');
          return res.send(500, err);
        }
        else{
          console.log('Query Success')
          console.log(products);
          if(products.length == 0){
            console.log('In product empty');
            startScraper(req, res);
```

```
        }
        else{
          console.log('In product not empty');
          res.render('products', {products});
        }

       }
      });
 }catch(err){
   console.log(err + ' in main async');
  }
}

async function startScraper(req, res){
 await scrapper(req.body.name);
  await Products.aggregate([{$match: { $text: { $search: req.body.name } }},{$project:
{productTitle : 1, productPrice : 1, productLink : 1, productImageLink : 1, productSite : 1, score:
{ $meta: "textScore" } }},{$match: { score: { $gt: 2 } }},{$project: {_id : 0, productTitle : 1,
productPrice : 1, productLink : 1, productImageLink : 1, productSite : 1}}], function(err,
products){
    if(err){
      console.log('error');
      return res.send(500, err);
    }
     res.render('products', {products});

  })

};


async function scrapper(searchString){
try{
  //Function to search the database for data use await---
  var browser = await puppeteer.launch({headless : true});
    let walmartResult = await scrapeWalmart(browser, searchString);

 await updateDatabase(walmartResult);
 await browser.close();

}catch(err){
  console.log(err);
}
}


//Amazon Scrapper
```

```
async function scrapeAmazon(browser, searchString){
  try{
    console.log('Scrapping Amazon.com');
    var productArrayAmazon = new Array();
    var basePage = basePages['amazon'];
    // Initiate the Puppeteer browser

    var page = await browser.newPage();
    // Go to the website page and wait for it to load

    await page.setRequestInterception(true);
    await page.on('request', (req) => {
      if(req.resourceType() == 'stylesheet' || req.resourceType() == 'font' || req.resourceType() ==
'image'){
          req.abort();
      }
      else {
          req.continue();
      }
    });


    await page.goto(basePage, { waitUntil: 'networkidle2', timeout : 0});
    //await page.waitForSelector('#twotabsearchtextbox');
    //Search for search box and enter product name
    await page.type('#twotabsearchtextbox', searchString);
    await page.click('.nav-search-submit > input:nth-child(2)');
    //Wait for page to load with results
    await page.waitForSelector('span.a-size-medium');
    // Run javascript inside of the page
    let data = await page.evaluate((searchString) => {
     var linkList = new Array();
      const list = document.querySelectorAll('span.a-size-medium');

        for(var i = 0; i < list.length; i++){
        //if((titleString.toLocaleLowerCase).startsWith())  {
          var flag = true;
          var string = list[i].innerText.toLowerCase();
          for(str of searchString.split(' ')){
              if(string.includes(str)){
                ;
              }
              else{
                flag = false;
                break;
```

```
            }
          }
          if(flag)
            linkList.push(list[i].parentElement.getAttribute('href'));
        }
      return linkList;
    }, searchString);
    console.log(data);
    for(const link of data){
      if(link == null)
        continue;
      var page = await browser.newPage();
      var pageToVisit = basePage + link;
      console.log('Visiting page : ' + pageToVisit);
      //await page.click('a[href =' + '\"' + link + '\"' + ']', { waitUntil: 'networkidle0', timeout : 0} );


      await page.setRequestInterception(true);

      await page.on('request', (req) => {
          if(req.resourceType() == 'stylesheet' || req.resourceType() == 'font' || req.resourceType()
== 'image'){
              req.abort();
          }
          else {
              req.continue();
          }
      });

      await page.goto(pageToVisit/*, { waitUntil: 'networkidle2', timeout : 0}*/);
      await page.waitForSelector('#landingImage');
      console.log('Evaluating product page');
      let pageData = await page.evaluate((pageToVisit) =>{
        var productTitle = document.querySelector('span#productTitle').innerText;
        var productPrice = document.querySelector('#priceblock_ourprice').innerHTML;
        var imageLink = "Product
Image";//document.querySelector('#landingImage').getAttribute('src');
        var productLink = pageToVisit;
        var productSite = 'Amazon';
        //console.log('Returning scraped data');
        return productTitle + '\n' + productPrice + '\n' + imageLink + '\n' + productLink + '\n' +
productSite;
      }, pageToVisit);
      await productArrayAmazon.push(createDataObject(pageData));
      await page.close();
  }
```

```
    console.log('Done');
}catch(err){
  console.log(err + ' In amazon scrapping');
}finally{
  return productArrayAmazon;
}

  };

//Scrape Walmart

async function scrapeWalmart(browser, searchString){

  try{
    console.log('Scrapping Walmart.com');
  var productArrayWalmart = new Array();
  var basePage = basePages['walmart'];
  //Initiate the Puppeteer browser
  //var browser = await puppeteer.launch({headless : true});
  var page = await browser.newPage();
  // Go to the IMDB Movie page and wait for it to load

  await page.setRequestInterception(true);

  await page.on('request', (req) => {
      if(req.resourceType() == 'stylesheet' || req.resourceType() == 'font' || req.resourceType() ==
'image'){
          req.abort();
      }
      else {
          req.continue();
      }
  });


  await page.goto(basePage, { waitUntil: 'networkidle2', timeout : 0});
 await page.type('#global-search-input', searchString);
  await page.click('#global-search-submit > span > span');
  await page.waitForSelector('a.product-title-link');
  // Run javascript inside of the page
  //basePage = page.url();

  let data = await page.evaluate((searchString) => {
    console.log('In page evaluate');
   var linkList = new Array();
    const list = document.querySelectorAll('a.product-title-link');
    for(var i = 0; i < list.length; i++){
```

```
    //if((titleString.toLocaleLowerCase).startsWith())  {
      var flag = true;
      var string = list[i].innerText.toLowerCase();
      for(str of searchString.split(' ')){
         if(string.includes(str)){
           ;
         }
         else{
           flag = false;
           break;
         }
      }
      if(flag)
        linkList.push(list[i].parentElement.getAttribute('href'));
    }
  return linkList;
  }, searchString);
  console.log(data);
  for(const link of data){
    if(link == null)
    continue;


    var page = await browser.newPage();
    var pageToVisit = basePage + link;

    await page.setRequestInterception(true);

    await page.on('request', (req) => {
       if(req.resourceType() == 'stylesheet' || req.resourceType() == 'font' || req.resourceType() ==
'image'){
          req.abort();
       }
       else {
          req.continue();
       }
    });



    await page.goto(pageToVisit/*, { waitUntil: 'networkidle0', timeout : 0}*/);
    await page.waitForSelector('body > div.js-content > div > div > div.js-body-content > div >
div.atf-content > div > div.atf-content > div > div > div > div > div.Grid > div.Grid > div:nth-
child(1) > div.product-image-carousel-container.product-image-mweb-container > div > div >
div > div > div.prod-hero-image > button > span > div.hover-zoom-container > div.hover-zoom-
hero-image-container > img');
    console.log('Evaluating product page');
```

```
    let pageData = await page.evaluate((pageToVisit) =>{
      console.log('Reading javascript');
      var productTitle = document.querySelector('h1').innerText;
      var productPrice = document.querySelector('.price-characteristic').innerText;
      var imageLink = "Product Image";//document.querySelector('body > div.js-content > div > div > div.js-body-content > div > div.atf-content > div > div.atf-content > div > div > div > div > div.Grid > div.Grid > div:nth-child(1) > div.product-image-carousel-container.product-image-mweb-container > div > div > div > div > div.prod-hero-image > button > span > div.hover-zoom-container > div.hover-zoom-hero-image-container > img').getAttribute('src');
      var productLink = pageToVisit;
      var productSite = 'Walmart';
      console.log('Returning scraped data');
      return productTitle + '\n' + productPrice + '\n' + imageLink + '\n' + productLink + '\n' + productSite;
      // console.log(product);
      //return product;
    }, pageToVisit);


    productArrayWalmart.push(createDataObject(pageData));
   await page.close();
}
  console.log('Done');
}catch(err){
  console.log(err + ' In Scrapping Walmart');
}
finally{
  return productArrayWalmart;
}

};


/*Ebay code*/

async function scrapeEbay(browser, searchString){
  try{
    console.log('Scrapping Ebay.com');
  var productArrayEbay = new Array();
  var basePage = basePages['ebay'];
  //Initiate the Puppeteer browser
  //var browser = await puppeteer.launch({headless : true});
  var page = await browser.newPage();


  await page.setRequestInterception(true);
```

```
  await page.on('request', (req) => {
    if(req.resourceType() == 'stylesheet' || req.resourceType() == 'font' || req.resourceType() ==
'image'){
        req.abort();
    }
    else {
        req.continue();
    }
  });


  await page.goto(basePage, { waitUntil: 'networkidle2', timeout : 0});
 await page.type('#gh-ac', searchString);
  await page.click('#gh-btn');
  await page.waitForSelector('span.BOLD');
  // Run javascript inside of the page

  let data = await page.evaluate((searchString) => {
  console.log('In page evaluate');
 const list = document.querySelectorAll('h3');
  var linkList = new Array();
  for(var i = 0; i < list.length; i++){
    //if((titleString.toLocaleLowerCase).startsWith())  {
      var flag = true;
      var string = list[i].innerText.toLowerCase();
      for(str of searchString.split(' ')){
          if(string.includes(str)){
            ;
          }
          else{
            flag = false;
            break;
          }
      }
      if(flag)
        linkList.push(list[i].parentElement.getAttribute('href'));
    }
    return linkList;
}, searchString);
 for(const link of data){
    if(link == null)
    continue;

    //await visitLinks(link, basePage);
  var page = await browser.newPage();
  var pageToVisit = link;
```

```
  //console.log('Visiting page : ' + pageToVisit);


  await page.setRequestInterception(true);

  await page.on('request', (req) => {
      if(req.resourceType() == 'stylesheet' || req.resourceType() == 'font' || req.resourceType() ==
'image'){
          req.abort();
      }
      else {
          req.continue();
      }
  });


  await page.goto(pageToVisit/*, { waitUntil: 'networkidle0', timeout : 0}*/);
  await page.waitForSelector('#icImg');


    let pageData = await page.evaluate((pageToVisit) =>{
    console.log('Reading javascript');
    var productTitles = document.querySelector('h1').cloneNode(true);
    productTitles.removeChild(productTitles.firstChild);
     var productTitle = productTitles.innerText;
    var productPrice = document.querySelector('span.notranslate').innerHTML;
    var imageLink = "Product Image";//document.querySelector('#icImg').getAttribute('src');
    var productLink = pageToVisit;
    var productSite = 'Ebay';

    console.log('Returning scraped data');
    return productTitle + '\n' + productPrice + '\n' + imageLink + '\n' + productLink + '\n' +
productSite + '\n';
  }, pageToVisit);

  //console.log(pageData);
  productArrayEbay.push(createDataObject(pageData));
  await page.close();
  }

}catch(err){
  console.log(err + 'in scraping ebay');
}
finally{
  console.log('Done');
  return productArrayEbay;
```

```
}
};



function updateDatabase(productArray){
  console.log(productArray);
  Products.collection.insertMany(productArray, function(err){
  if(err){
    console.log('Unable to save the product to database');
  }
  else{
    console.log('Product saved successfully');
  }
 });
}



function createDataObject(pageData){
  var productDetails = pageData.split("\n");
  console.log(productDetails);
  var prodObj = {
    productTitle : productDetails[0],
    productPrice : productDetails[1],
    productLink : productDetails[3],
    productImageLink : productDetails[2],
    productSite : productDetails[4]
  };

  return prodObj;

}
```

Database file – products.js

```
const mongoose = require('mongoose');

let productSchema = new mongoose.Schema({
    productTitle : String,
    productPrice : String,
    productLink : String,
    productImageLink : String,
    productSite : String
});
```

```
//creating index on productTitle field.
productSchema.index({productTitle : 'text'});
let Products = mongoose.model('Products', productSchema);

module.exports = Products;
```

db.js – Database config file

```
const mongoose = require('mongoose');

const MONGO_USERNAME = "admin";
const MONGO_PWD = "admin";
const MONGO_HOSTNAME = "127.0.0.1";
const MONGO_PORT = "27017";
const MONGO_DB = "admin";


const url = "mongodb://admin:admin@127.0.0.1:27017/admin";
//const url =
"mongodb://${MONGO_USERNAME}:${MONGO_PWD}@${MONGO_HOSTNAME}:${M
ONGO_PORT}/${MONGO_DB}";
mongoose.connect(url, {useNewUrlParser : true,  useFindAndModify: false });
```

app.js

```
const express = require('express');
//const path = require('path');
const bodyParser = require('body-parser');
const routes = require('./routes/routes');
const db = require('./db');
var app = express();
var port = 3000;

    const path = __dirname + '/views';
    //app.engine('html', require('ejs').renderFile);
    app.set('view engine', 'ejs');
    app.use(express.urlencoded({ extended: true }));
    app.use(express.static(path));
    app.use(bodyParser.json());
    app.use(bodyParser.urlencoded({ extended: true }));
    app.use('/routes', routes);
    //app.use(express.static(__dirname + '/views'));

    app.listen(port, ()=>{
```

```
        console.log("Server listening on port - " + port);
})
```

Express.js – Routes file – routes.js

```
const express = require('express');
const router = express.Router();
const products = require('../controllers/server');

router.get('/', function(req, res){
   products.index(req,res);
});


 router.post("/getData", (request, response)=>{
    console.log('In routes ' + request.body.name);
    products.getData(request, response);
    //products.push(product)
    //response.render('products',{products});
 });

 module.exports = router;
```

View files -

products.ejs

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title></title>
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.css" />
  <!-- <link href='./css/styles.css' rel="stylesheet" type = 'text/css'/> -->
  <style>
        .alert {
          padding: 10px;
          background-color: #d3d3d3;
          /* Red */
          color: #000000;
          text-align: center;
          margin: 0 auto;
          display: none;
```

```
}

/* The close button */
.closebtn {
  margin-left: 15px;
  color: white;
  font-weight: bold;
  float: right;
  font-size: 22px;
  line-height: 20px;
  cursor: pointer;
  transition: 0.3s;
}

/* When moving the mouse over the close button */
.closebtn:hover {
  color: black;
}

.btn-primary {
  background: #000;
  color: #fff;
}

.card {
  flex: auto;
  box-sizing: border-box;
  margin: 1rem 0.25em;
}

.card-body:hover {
  background-color: #d3d3d3;
  color: black;
}

.proceed {
  background: #000;
  color: #fff;
  border: none;
}

.card-body:hover .btn-primary {
  visibility: visible;
}

.card-body {
```

```
        background: #000;
        color: #fff;
        border-radius: 10px;
      }

      .main {
        background-color: #eee;
      }

      .primary {
        position: relative;
        visibility: hidden;
      }
    </style>
</head>

<body class="main">
      <div className="col-xs-12" style="width: 100%; height: 100%">
        <div>
          <!-- Navigation -->
          <nav style="background-color: #000" class="navbar navbar-expand-md sticky-top">
            <div class="container-fluid">
              <div id="path"
                style="color: #7EC0EE; font-weight: 700; font-size: 20px; margin: 0 auto; font-
family:sans-serif;">
                <h1>Comparify.com</h1>
              </div>
            </div>
          </nav>
        </div>
      </div>
      <div class="row" style="width: 75%; margin: 0 auto; background:#fff; height: 100%;
display:flex; flex-wrap: wrap;
       padding:50px;">

      <% products.forEach(function(product){ %>

              <div class=" container" style=" width: 18rem; min-height: 10px; margin-
top:30px;">
                  <a href="#"></a>
                  <div class="card-body">
                  <div class="row">
                     <img src="<%=product.productImageLink%> " alt = "product image"/>
                     <p>Product Name : <%=product.productTitle%></p>
                     <p>Price : <%=product.productPrice%></p>
```

```
                    <p>Product link : <a href = " <%=product.productLink%> ">Buy
now</a></p>
                    <p>Product site : <%=product.productSite%></p>
                  </div>
                  </div>
                </div>
              <% }); %>

      </div>

<!--                 <h5 style="margin:0 auto;">Check Out</h5>
                <p style="margin:0 auto; padding: 20px; text-align: center;">
                  Clone the desired repository by providing manifest file
                </p>
                <a href="#" class="btn btn-primary proceed" data-toggle="modal" data-
target="#myModal2"
                  style="margin: 0 auto;">Proceed</a>-->

          <script src="https://code.jquery.com/jquery-3.3.1.js"></script>
          <script src="/js/home.js"></script>
          <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
          <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"></script>
          <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
          <script src="https://use.fontawesome.com/releases/v5.0.8/js/all.js"></script>
        </body>

        </html>

<!--
<!doctype html>

<html>
<head>
  <title>Welcome to 911 mobiles</title>
</head>
<body>
  <main>
    <header>
    </header>
    <article>
          <% products.forEach(function(product){ %>
            <section>
              <img src="<%=product.productImageLink%> " alt = "product image"/>
              <p>Product Name : <%=product.productTitle%></p>
```

```
                <p>Price : <%=product.productPrice%></p>
                <p>Product link : <a href = " <%=product.productLink%> ">Buy now</a></p>
                <p>Product site : <%=product.productSite%></p>
            </section>
        <% }); %>
    </article>
    <footer>
    </footer>
  </main>
</body>
 </html> -->
```

Index.html

```
<!doctype html>

<html>
<head>
  <title>Welcome to 911 mobiles</title>
</head>
<body>

  <main>
    <header>
    </header>
    <article>
          <form class="form-inline md-form mr-auto mb-4" method = "POST",
action="routes/getData">
              <input class="form-control mr-sm-2" type="text" placeholder="Enter product to
search" name = "name" aria-label="Search">
              <button class="btn btn-elegant btn-rounded btn-sm my-0"
type="submit">Search</button>
            </form>
    </article>
    <footer>
    </footer>
  </main>
</body>
</html>
```

REFERENCES

[1] Shikha Mahajan, Nikhit Kumar “A Web Scraping Approach in Node.js ” 2015 *International Journal of Science, Engineering and Technology Research (IJSETR)*

[2] Sanjay Kumar Malik1 , SAM Rizvi2 “Information Extraction using Web Usage Mining, Web Scrapping and Semantic Annotation” 2011 *International Conference on Computational Intelligence and Communication Systems.*

[3] Aleksi Allilio “automated web store product scraping using node.js” 2015 *Tampere University of Technology*.

[4] https://mongoosejs.com/docs/guide.html - Mongoose documentation.

[5] https://pptr.dev/ - Puppeteer Library documentation.