**A**

***PROJECT REPORT***

*on*

**Product Automation**

*Submitted in partial fulfilment of the requirements for the degree of*

**BACHELOR OF TECHNOLOGY**

****

Session: - Jan-June 2023

Submitted by

Palash Solanki (19ETCEC011)

8th semester, ECE

Under Guidance of

Ekansh Jain

The Developer Company

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**TECHNO INDIA NJR INSTITUTE OF TECHNOLOGY, UDAIPUR-313001**

**MAY - 2023**

**A**

***PROJECT REPORT***

*on*

**Product Automation**

*Submitted in partial fulfilment of the requirements for the degree of*

**BACHELOR OF TECHNOLOGY**



**Session: - Jan-June 2023**

Submitted by

Palash Solanki (19ETCEC)

8th Semester, ECE

Under Guidance of

Ekansh Jain

The Developer Company

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**TECHNO INDIA NJR INSTITUTE OF TECHNOLOGY, UDAIPUR-313001**

**MAY - 2023**



Department of Electronics and Communication Engineering

Techno India NJR Institute of Technology, Udaipur-313001

**Certificate**

This is to certify that project work titled PRODUCT AUTOMATION by Palash Solanki is successfully carried out in the Department of Electronics and Communication Engineering, TINJRIT and the report is approved for submission in the partial fulfillment of the requirements for award of degree of Bachelor of Technology in Electronics and Communication.

Ekansh Jain Dr. Vivek Jain

The developer Company Dept. of E.C.E TINJRIT, Udaipur

 Date...................... Date......................



Department of Electronics and Communication Engineering

Techno India NJR Institute of Technology, Udaipur-313001

**Examiner Certificate**

This is to certify that the following student

**Palash Solanki**

of final year B.Tech. (Electronics & Communication Engineering), was examined for the project work titled

***Product Automation***

during the academic year 2022 – 2023 at Techno India NJR Institute of Technology, Udaipur

**Remarks:**

**Date:**

 Signature Signature

(**Internal Examiner**) (**External Examiner**)

Name :- ……………………… Name :- ………………………

Designation:- ……………….. Designation:- ………………..

Department: - ………………. Department: - ……………….

Organization:- ……………… Organization:- ………………

**Preface**

Web scraping is the process of extracting data from websites automatically using software tools. It involves fetching web pages, parsing the HTML content, and extracting the desired information for further analysis or storage. Several tools are commonly used in web scraping, including Axios, Cheerio, Puppeteer, Fetch, and RegEx. Let's take a closer look at each of these tools and their roles in the web scraping process:

1. Axios: Axios is a popular JavaScript library used for making HTTP requests from a web browser or Node.js. It simplifies the process of fetching web pages by providing a simple and intuitive API for sending HTTP requests and handling responses. Axios can be used to download web pages, which can then be parsed and scraped using other tools.
2. Cheerio: Cheerio is a fast and flexible library that implements a jQuery-like syntax for parsing and manipulating HTML or XML documents. It allows you to traverse and manipulate the parsed document using familiar CSS selectors, making it easy to extract specific elements or data from the web page.
3. Puppeteer: Puppeteer is a Node.js library developed by Google that provides a high-level API for controlling headless Chrome or Chromium browsers. It allows you to automate browser actions, such as navigating to web pages, interacting with elements, and extracting data. Puppeteer is a powerful tool for web scraping that can handle complex scenarios, including JavaScript-rendered pages.
4. RegEx (Regular Expressions): Regular expressions are a powerful tool for pattern matching and text manipulation. They allow you to define a pattern and search for matches within a string or document..

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

TECHNO INDIA NJR INSTITUTE OF TECHNOLOGY, UDAIPUR-313001

**ACKNOWLEDGMENT**

We take this opportunity to record our sincere thanks to all who helped us to successfully complete this work. Firstly, We are grateful to our **supervisor Ekansh Jain** for his invaluable guidance and constant encouragement, support and most importantly for giving us the opportunity to carry out this work.

We would like to express our deepest sense of gratitude and humble regards to our

**Head of Department Dr. vivek Jain** for giving invariable encouragement in our endeavors and providing necessary facility for the same. Also a sincere thanks to all faculty members of ECE, TINJRIT for their help in the project directly or indirectly.

Finally, We would like to thank my friends for their support and discussions that have proved very valuable for us. We are indebted to our parents for providing constant support, love and encouragement. We thank them for the sacrifices they made so that we could grow up in a learning environment. They have always stood by us in everything we have done, providing constant support, encouragement and love

**Palash Solanki (19ETCEC011)**

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

TECHNO INDIA NJR INSTITUTE OF TECHNOLOGY, UDAIPUR-313001

**ABSTRACT**

Web scraping has become an essential technique in extracting data from websites for various purposes, such as data analysis, research, and automation. This abstract provides an overview of web scraping methodologies using popular tools like Axios, Cheerio, Puppeteer, and Postman.

Axios is a widely-used JavaScript library for making HTTP requests from a web browser or Node.js. It simplifies the process of sending HTTP requests and handling responses. When combined with Cheerio, a fast and flexible HTML parsing library, Axios enables developers to scrape data from HTML documents efficiently. Cheerio provides a jQuery-like syntax for traversing and manipulating the HTML structure, making it easier to extract specific data from web pages.

Puppeteer is a powerful Node.js library developed by Google that allows for browser automation. It provides a high-level API to control and interact with the Chrome or Chromium browser programmatically. With Puppeteer, developers can navigate websites, fill out forms, click buttons, and extract data from dynamically generated content. Its headless mode allows for running browser automation in a headless environment, making it ideal for web scraping tasks.

This abstract outlines the benefits and applications of using Axios, Cheerio, Puppeteer, and Postman in web scraping projects. It emphasizes their strengths in different scenarios and provides insights into their integration and usage. Furthermore, it highlights the importance of adhering to ethical considerations and legal guidelines when performing web scraping to ensure the responsible and respectful use of data.

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Abstract** |  | 7 |
| **Contents** |  | 8 |
| **Chapter 1**  | INTRODUCTION | 9 |
|  | * 1. Applications
 | 10 |
|  | 1.2 Tools Used  | 11 |
| **Chapter 2** | Description | 12 |
|  | * 1. Definition And Properties of Web Scraping
 |  13 |
|  | * 1. Applications of Axios
 | 14 |
|  | * 1. Applications of Cheerio
 | 15 |

|  |  |
| --- | --- |
| **Chapter 3** | Working 163.1 Axios 163.2 Cheerio 173.3 Puppeteer 18 |

**Appendices**

**Conclusion**

**Chapter 1: INTRODUCTION**

Product automation refers to the process of using technology and software to automate various tasks and processes involved in the production and management of goods and services. It involves the use of machines, robots, and computer systems to streamline and optimize production, reduce manual labor, and enhance overall efficiency.

Product automation can be applied in various industries, including manufacturing, logistics, agriculture, healthcare, and more. It involves automating repetitive tasks such as assembly line operations, inventory management, quality control, packaging, and distribution. By automating these processes, companies can increase productivity, reduce costs, improve accuracy, and accelerate time-to-market.

Web scraping has become increasingly popular due to the vast amount of data available on the internet and the need to extract and analyze this data for various purposes. It has applications in e-commerce, market research, data mining, price comparison, content aggregation, lead generation, and many other fields.

Web scraping can be performed using programming languages like Python, along with libraries such as BeautifulSoup and Scrapy. These tools allow developers to navigate through web pages, locate specific elements, and extract relevant data, such as text, images, links, tables, and more.

* 1. Applications
		1. Price Monitoring: Web scraping allows you to track the prices of products across different websites. You can set up automated scripts to scrape the prices regularly and notify you when there are changes. This information can be used for competitive analysis, dynamic pricing strategies, or identifying pricing trends
		2. Stock Availability: If you need to monitor the availability of certain products on different e-commerce platforms, web scraping can help. By scraping websites for stock information, you can automate the process of checking if a product is in stock or out of stock.
		3. Product Information Extraction: Web scraping allows you to gather detailed information about products, such as descriptions, specifications, images, and features. This data can be used to build a comprehensive product catalog, compare products across multiple platforms, or generate content for your own website or application
		4. Data Aggregation: Web scraping can automate the process of collecting data from different sources and aggregating it into a unified format. For example, if you sell products across multiple e-commerce platforms, you can scrape the relevant data from each platform and consolidate it into a single database. This allows you to analyze and manage your product data more efficiently.
	2. Tools Used
* Postman for API calling
* Cheerio
* Axios
* Puppeteer
* RegExr
* File System
* Lodash

These are just a few examples of tools and libraries used in web scraping. The choice of tools depends on your programming language preference, the complexity of the task, and the specific requirements of your scraping project.

**Chapter 2: DESCRIPTION**

Product automation refers to the use of technology and software to streamline and automate various tasks and processes in a product's lifecycle. It involves leveraging tools and systems to minimize manual intervention and increase efficiency. Automation can be applied to different stages, such as product development, testing, deployment, and maintenance. By automating repetitive and time-consuming tasks, product automation allows teams to focus on higher-value activities and accelerates the overall product delivery.

Web scraping, on the other hand, is a technique used to extract data from websites automatically. It involves writing scripts or using specialized software to navigate through web pages, gather information, and store it in a structured format for analysis or further use. Web scraping enables businesses and individuals to extract large volumes of data from multiple sources, such as e-commerce websites, social media platforms, and news sites. The extracted data can be utilized for various purposes, including market research, competitive analysis, price monitoring, content aggregation, and more.

Both product automation and web scraping share the goal of increasing efficiency and productivity. Product automation streamlines internal processes, reduces errors, and ensures consistency, allowing teams to deliver products faster and with higher quality. It eliminates manual work, repetitive tasks, and human errors, leading to improved efficiency and cost savings. On the other hand, web scraping automates the process of data collection from websites, eliminating the need for manual copy-pasting or monitoring. It enables businesses to gather valuable insights from a vast amount of data in a shorter time, enabling informed decision-making and gaining a competitive edge.

Overall, product automation and web scraping are powerful techniques that leverage

2.1 Definition And Properties of Web Scraping

1. Automated data extraction: Web scraping is an automated process that eliminates the need for manual data collection. It enables the retrieval of large amounts of data from multiple web pages or websites quickly and efficiently.
2. Parsing HTML or structured content: Web scraping typically involves parsing the HTML structure of web pages or working with structured content formats like JSON or XML. The scraper analyzes the structure and identifies the relevant elements or data to extract
3. Data extraction methods: Web scraping can employ various methods to extract data, including text matching, XPath queries, CSS selectors, or regular expressions. These techniques allow the scraper to locate specific elements, such as headings, tables, lists, or data within specific tags.
4. Integration and analysis: The extracted data from web scraping can be further processed, analyzed, or integrated into other systems or applications. This allows for a wide range of use cases, such as market research, price comparison, data aggregation, sentiment analysis, or machine learning training datasets.
5. Handling dynamic content: Many modern websites use dynamic content loading techniques, such as AJAX or JavaScript, to update or modify the page after it loads. Web scrapers may need to handle such dynamic content by using headless browsers or interacting with APIs to retrieve the desired data.

2.2 Applications of Axios

1. AJAX requests: Axios allows you to make asynchronous HTTP requests from web browsers, commonly known as AJAX (Asynchronous JavaScript and XML) requests. You can use Axios to fetch data from APIs, send data to a server, or interact with any HTTP-based service.
2. RESTful API interactions: Axios simplifies the process of interacting with RESTful APIs. You can use Axios to send GET, POST, PUT, DELETE, and other HTTP requests to communicate with APIs and retrieve or modify data
3. Promises and async/await: Axios is built on top of JavaScript promises, making it easy to work with asynchronous operations. It also supports the use of async/await, which provides a more readable and synchronous-like coding style when making HTTP requests.
4. Handling response data: Axios provides various methods for handling response data, including JSON parsing, error handling, and interceptors. You can easily extract and manipulate the data received from the server in a structured format.
	1. Application of Cheerio
		1. Web Scraping: Cheerio is often used for web scraping tasks where you need to extract specific data from HTML pages. You can use Cheerio to navigate the HTML structure, select elements based on their attributes or content, and extract the desired information.
		2. Data Extraction: With Cheerio, you can easily extract data from HTML documents and convert it into a structured format such as JSON or CSV. You can select elements, access their attributes or text content, and extract the required data for further processing or analysis.
		3. Web Testing: Cheerio can be used for testing web applications by parsing the HTML response and performing assertions on specific elements or data. You can simulate interactions, manipulate the DOM, and verify if the expected changes or behaviors occur.
		4. Server-side Rendering: Cheerio can be used in server-side rendering frameworks like Next.js or Nuxt.js to manipulate HTML content before it is sent to the client. This allows you to dynamically modify the rendered HTML based on the server-side data or perform optimizations for SEO purposes

**Chapter 3: WORKING**

 3.1 Working of Axios :-

 Axios is a popular JavaScript library used for making HTTP requests from web browsers or Node.js. It provides a simple and intuitive API for sending asynchronous requests to servers and handling the responses. While Axios itself does not specifically focus on product automation, it can be utilized as a part of automation workflows to interact with APIs or perform HTTP requests as needed.

1. Installation: To use Axios in your project, you need to install it first. You can do this by running the following command in your terminal or command prompt: npm install axios
2. Importing Axios: Once installed, you can import Axios into your JavaScript file using the require statement (for Node.js) or import statement (for modern JavaScript environments):
3. Making HTTP Requests: Axios provides several methods for making HTTP requests such as **axios.get**, **axios.post**, **axios.put**, **axios.delete**, etc. You can choose the appropriate method based on the type of request you need to send. For example, to make a GET request to retrieve data from a server
4. Sending Request Body: For POST, PUT, or PATCH requests, you can send data in the request body. You can pass the data as an object:
5. Configuring Axios: You can also configure Axios globally or per request basis by creating an instance of Axios and customizing its settings. This allows you to set defaults, intercept requests or responses, etc. For example:

These are the basic steps involved in using Axios for making HTTP requests. By incorporating Axios into your product automation workflows, you can interact with APIs, fetch data from servers, send data, and handle responses effectively.

3.2 Working of Cheerio

 Cheerio is a fast, flexible, and lightweight library for parsing and manipulating HTML documents in Node.js. It provides a jQuery-like interface for traversing and manipulating the HTML structure using familiar CSS selectors.

1. **Installation**: First, you need to install Cheerio in your Node.js project. You can do this by running the following command using npm (Node Package Manager):( npm install cheerio)
2. **Loading HTML**: Once Cheerio is installed, you can load an HTML document into Cheerio. You can pass the HTML string or a file path to the **cheerio.load()** function to create a Cheerio instance. For example:
3. **DOM Traversal**: Cheerio provides a range of methods that mimic jQuery's DOM traversal methods. You can select elements using CSS selectors and perform various operations on them. For example, to select all **<h1>** elements and get their text content:
4. Manipulating Elements: You can modify the HTML structure by adding, removing, or modifying elements. Cheerio provides methods like append(), remove(), attr(), addClass(), etc., to manipulate the elements. Here's an example of adding a new <div> element:
5. Serialization: Cheerio provides methods to serialize the modified HTML back into a string. You can use the $.html() method to retrieve the modified HTML content. For example:

Cheerio's simplicity and ease of use make it a popular choice for parsing and manipulating HTML in Node.js applications, especially for tasks like web scraping, data extraction, and automated testing.

3.3 Working of Puppeteer

 Puppeteer is a Node.js library developed by the Chrome team at Google, which provides a high-level API for automating and controlling headless Chrome or Chromium browsers. It allows you to interact with web pages, perform automated tasks, and scrape data from websites.

1. Installation: To use Puppeteer, you need to install it as a dependency in your Node.js project. You can do this using npm or yarn.
2. Launching a Browser: Puppeteer allows you to launch a headless Chrome or Chromium browser instance. You can specify various options during the launch, such as the executable path, user agent, viewport size, etc.
3. Opening a Page: Once the browser is launched, you can open a new page/tab within it. The page object represents the web page you want to interact with.
4. Navigation: Puppeteer provides methods to navigate to different URLs, reload the page, go forward or backward in history, or wait for navigation to complete.
5. Extracting Data: Puppeteer allows you to extract data from web pages by evaluating JavaScript code within the context of the page. You can access the page's DOM, query elements, get attribute values, and retrieve text content.
6. Taking Screenshots: Puppeteer provides methods to capture screenshots of the web page. You can take full-page screenshots, capture specific elements, or define custom viewport sizes.
7. Authentication and Security: Puppeteer provides functionality to handle authentication prompts, set cookies, manipulate the browser's local storage, and modify request headers. It also supports bypassing SSL certificate errors.
8. Closing the Browser: Once you have finished your automation tasks, you can close the browser instance using the provided API.

Puppeteer offers a wide range of features and flexibility for automating browser interactions and web scraping. It is widely used for tasks such as website testing, data scraping, automated form filling, and generating screenshots or PDFs of web pages.

**APPENDICES**

import { eachOfLimit } from 'async'

import axios from 'axios'

import { load } from 'cheerio'

import { readFile, writeFile } from 'fs/promises'

const headers = {

  'User-Agent':

    'Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0 Safari/537.36',

  cookie:

    ''

}

const formatString = (*str*: string) =>

  *str*

    .replaceAll('\t', ' ')

    .replaceAll('\n', ' ')

    .replaceAll('&nbsp;', ' ')

    .trim()

    .replaceAll(/[\s]{2,}/g, '<->')

    .replaceAll('<->', ' ')

const inStockMapping = [

  'special order',

  'in network',

  'in stock',

  'limited quantities',

  'ships directly',

  'remaining on allocation',

  'cold chain items'

]

const scrapData = async () => {

  const urls: {

    id: string

  }[] = JSON.parse(await readFile('mcKessonIds.json', 'utf-8'))

  const finalData: any[] = []

  await eachOfLimit(urls, 10, async (*url*, *i2*) => {

    const i = parseInt(*i2*.toString())

    console.log(`${i + 1}/${urls.length}`)

    try {

      const res = await fetch(

        'XYZ' + *url*.id,

        {

          headers

        }

      )

      const html = await res.text()

      const $ = load(html)

      let flag = false

      $('.item-selectuom > option').each((*\_*, *el*) => {

        flag = true

        const splitted = formatString($(*el*).text().trim()).split('$')

        const uom = splitted[0].trim().split('/')[0]

        const price = splitted[1].trim()

        let id = formatString(

          $('#pageContent > section > ul:nth-child(1) > li.dontprint').text()

        )

        if (id.includes('Ordered')) id = id.split('Ordered')[0]

        const stockMessage = formatString($('.status-messaging').text().trim())

        const stockStatus = inStockMapping.find((*str*: string) =>

          stockMessage.toLowerCase().includes(*str*)

        )

          ? 'Available'

          : 'Out of Stock'

        const categories: string[] = []

        $('.breadcrumbs.dontprint > span').each((*\_*, *el*) => {

          const c = formatString($(*el*).text().replaceAll('>', '').trim())

          if (c) categories.push(c)

        })

        const images: string[] = []

        $('#imgZoom > img').each((*\_*, *el*) => {

          const src = $(*el*).attr('src')

          if (src) images.push(src)

        })

        const options: any[] = []

        $(

          '#pageContent > div.page-flex-row.row3col > div:nth-child(2) > section > h4'

        ).each((*i*, *el*) => {

          if (*i* < 1) return

          options.push({

            name: $(*el*).text().trim(),

            value: ''

          })

        })

        $(

          '#pageContent > div.page-flex-row.row3col > div:nth-child(2) > section > ul > li > a.current'

        ).each((*i*, *el*) => {

          if (options[*i*]) options[*i*].value = $(*el*).text().trim()

        })

        const obj = {

          url: 'XYZ' + *url*.id,

          id,

          mfr: formatString(

            $('#pageContent > section > ul:nth-child(1) > li:nth-child(3)')

              .text()

              .trim()

          ),

          name: $('h1').text().trim(),

          brand: formatString(

            $('#pageContent > section > ul:nth-child(1) > li:nth-child(2)')

              .text()

              .trim()

          ),

          uom,

          uomToEach: splitted[0].trim().split('/')[1],

          price,

          stockMessage,

          stockStatus

          categories: categories.join(' > '),

          images: images.join(','),

          features: $('#itemFeatures').html(),

          specifications: $('#specs > table').html(),

          options

        }

        finalData.push(obj)

      })

      if (!flag)

        finalData.push({

          url,

          error: 'No price data found'

        })

    } catch (err) {

      console.error(err)

      finalData.push({

        url,

        error: true

      })

    }

  })

  await writeCsv('', finalData)

}

**CONCLUSION**

Webscraping refers to the process of extracting data from websites by using automated scripts or software. It allows for the collection of vast amounts of data from multiple sources quickly and efficiently. Webscraping has numerous applications, including market research, data analysis, price comparison, sentiment analysis, and content aggregation. It enables businesses to gather valuable insights, make informed decisions, and stay competitive in the digital age.

Automation, on the other hand, involves the use of software or scripts to perform repetitive tasks automatically. It saves time, reduces errors, and increases productivity. Automation can be applied to various activities, such as data entry, report generation, content publishing, social media management, and customer support. By automating routine processes, businesses can streamline operations, focus on more complex tasks, and enhance overall efficiency.

In summary, webscraping and automation offer tremendous opportunities for businesses and individuals to extract data, automate processes, and enhance productivity. When used responsibly and within legal boundaries, these tools can provide valuable insights, improve efficiency, and contribute to the growth and success of various industries.