MARRI LAXMAN REDDY GROUP OF INSTITUTIONS MLR INSTITUTE OF TECHNOLOGY

(UGC AUTONOMOUS)

Laxman Reddy Avenue, Dundigal, Hyderabad - 500 043, Telangana, India

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (DATA SCIENCE)

A REPORT

ON

”WEB DATA EXTRACTION FROM WEBSITES”

B.Tech (CSE - DATA SCIENCE)

SUBMITTED BY

KOLLURU VENKATA ADITHYA (22R21A6792)
KADAPA TAUFEEQ UMAR (22R21A6786)
BOLLEDU VARUN (22R21A6768)

UNDER THE GUIDANCE OF

Ms.K.SRINIJA

(Academic Year: 2023-2024)

MARRI LAXMAN REDDY GROUP OF INSTITUTIONS MLR INSTITUTE OF TECHNOLOGY (UGC AUTONOMOUS)

Laxman Reddy Avenue, Dundigal, Hyderabad - 500 043, Telangana, India

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (DATA SCIENCE)

Certificate

This is to certify that project entitled

"WEB DATA EXTRACTION FROM WEBSITES"

has been completed by
KOLLURU VENKATA ADITHYA (22R21A6792)
KADAPA TAUFEEQ UMAR (22R21A6786)
BOLLEDU VARUN (22R21A6768)

of B.Tech CSE(DS), II Year, II Semester of academic year 2023-2024 in partial fulfillment of the Second Year of Bachelor degree in "Computer Science & Engineering (Data Science)" as prescribed by the MLR Institute of Technology.

Ms.K.Srinija	**Dr.Chiranjeevi Manike**
Supervisor	**H.O.D**

External Examiner

ACKNOWLEDGEMENT

It gives me great pleasure and satisfaction in presenting this mini project on "WEB DATA EXTRACTION FROM WEBSITES".

I would like to take this opportunity to express my heartfelt gratitude to all those who have contributed to the successful completion of this project

I have furthermore to thank Department HOD, **Dr.Chiranjeevi Manike** *and* **Ms.K.Srinija** *to encourage me to go ahead and for continuous guidance.*

I would like to thank all those, who have directly or indirectly helped me for the completion of the work during this mini project.

KOLLURU VENKATA ADITHYA (22R21A6792)
KADAPA TAUFEEQ UMAR (22R21A6786)
BOLLEDU VARUN (22R21A6768)
B.Tech CSE (DS)

Contents

List of Figures

List of Tables

Abstract

Web scraping, a dynamic technique, has emerged as an indispensable tool for gathering structured information from websites and online sources. Web scraping encompasses a wide range of automated data extraction methods that enable users to access and collect data from websites and web applications. These techniques involve parsing the HTML, XML, or other structured data formats of web pages to extract specific information, such as text, images, links, and more. Web scraping plays a pivotal role in the digital age, offering unprecedented access to valuable data for research, business, and decisionmaking purposes. As the digital frontier continues to expand, web scraping remains a powerful tool for unlocking insights and opportunities from the vast online ecosystem. Web scraping should be conducted ethically and within legal boundaries. Some websites explicitly prohibit web scraping in their terms of service and scraping sensitive or personal data without consent is illegal in many jurisdictions. Web scraping is a powerful tool for collecting and utilizing data from the web. .

Chapter 1

INTRODUCTION

1.1 Introduction

Web scraping is a way to retrieve data from websites, many websites do not allow the users to save data for personal use surfing through the web. Manually copying and pasting the data is a tedious and time consuming task. Web Scraping automates the data extraction process from websites. The use of web scrapers is used to carry out this task. They automatically load the websites and extract data from them based on user requirements. The purpose of this project is to create a Web scraper that is capable of giving the required data in a structured format. The website will collect the information for research, monitoring prices, extracting information for analysis, or aggregating content for a website. The key is to automate the extraction of data from websites in a structured manner for further use or analysis. The usage of this web scraper can vary depending on the goals, but commonly we use it for monitoring website changes or extracting data for analysis. It helps save time compared to manual extraction.

1.2 Motivation

Web data extraction projects are often fueled by multifaceted objectives. Businesses leverage this process to gain crucial insights into market trends, competitive landscapes, and consumer behavior, empowering informed decision-making. Academic and professional researchers harness web data extraction to contribute to studies and comprehensive analysis. Automation of data collection optimizes workflows, enhancing operational efficiency across industries like finance, e-commerce, and healthcare. Extracted data can be seamlessly integrated into existing systems, offering tailored solutions or bolstering tool functionality. .

Chapter 2

PROBLEM STATEMENT

2.1 Problem Statement

In the current digital era, the internet hosts an immense volume of valuable data that presents both opportunities and challenges for businesses and researchers. A critical challenge faced is the efficient extraction and utilization of structured data from websites. Web data extraction involves the automated retrieval of specific data elements from web pages, essential for tasks such as market research, competitive analysis, and business intelligence. However, this process is hindered by technical complexities in developing robust scraping solutions, ensuring data accuracy and legality, and navigating ethical considerations. Addressing these challenges is crucial to harnessing the full potential of web data for informed decision-making and competitive advantage in the digital marketplace.

2.2 Explanation

In today's digital age, the vast amount of information available on the internet presents both opportunities and challenges for businesses and researchers alike. One critical challenge is the efficient extraction and utilization of structured data from websites. Web data extraction involves retrieving specific data elements from web pages, which can be used for various purposes such as market research, competitive analysis, and business intelligence.

2.3 Challenges

1. Technical Complexity:
    - Developing effective web scraping tools requires expertise in programming languages and familiarity with web technologies.
    - Websites often employ dynamic content loading and anti-scraping techniques, necessitating sophisticated parsing and data extraction methods.
2. Data Quality and Reliability:

- Ensuring the accuracy, completeness, and consistency of extracted data poses significant challenges.
- Websites frequently update their layouts and content structures, leading to potential data discrepancies or extraction errors.

3. Legal and Ethical Considerations:
    - The legality of web scraping varies across jurisdictions and can be influenced by website terms of service.
    - Ethical concerns arise regarding the privacy implications of collecting personal data and the appropriate use of scraped information.
4. Operational Challenges:
    - Scalability and reliability issues arise when scraping large volumes of data.
    - Websites may implement measures like CAPTCHA challenges or IP blocking to deter automated access, adding to operational complexity.

2.4 Objectives

1. Developing Efficient Web Scraping Techniques:
    - Create methodologies that ensure extracted data maintains high quality and reliability standards.
    - Implement advanced parsing and extraction methods to handle dynamic web content and anti-scraping measures effectively.
2. Addressing Legal and Ethical Considerations:
    - Ensure compliance with legal requirements across different jurisdictions, including adherence to website terms of service.
    - Uphold ethical standards in data collection, particularly regarding user privacy and the responsible use of scraped information.
3. Overcoming Technical Hurdles:
    - Develop scalable solutions capable of handling diverse web structures and large volumes of data.
    - Mitigate challenges posed by CAPTCHA challenges, IP blocking, and other anti-scraping measures through innovative technical approaches.
4. Empowering Business and Research Applications:
    - Enable businesses to leverage web data for comprehensive market insights, competitive analysis, and informed strategic decision-making.
    - Provide researchers with robust tools to extract and analyze data for academic and scientific purposes, fostering innovation and knowledge discovery.

Chapter 3

LITERATURE SURVEY

We conducted a thorough literature survey by reviewing existing systems for web data extraction from websites. Research papers, journals, and publications have also been referred to prepare this survey.David Mathew Thomas; and Sandeep Mathur

[3] This paper is mainly about how data isextracted using Python software from different websites that have different algorithms to extract the data. In this paper, there is a specified language Python, which determines what is themethodology of the language and how it is implemented.

Ayat Abodayeh; Reem Hejazi; WardNajjar; Leena Shihadeh; and Rabia Latif[7] proposed a new algorithm which is Beautiful Soup which is a simple and easiest algorithm for extracting the data from the website.

This form of extraction can extract whole data at a time. Chun Feng Lin; and Sheng Chih Yang[6] have proposed a system that mainly focuses on Taiwan stock market prediction in which the web scraping mechanism is used which is time a time extractor because every minute the strokes may change their price.

Through this, it is helpful to invest the money in the right stock

REFERENCES

- Choudhary, D., Sharma, M., Gupta, S. (2020). Web Data Extraction Techniques: A Review. International Journal of Computer Applications, 176(38), 1-7.
- Mitchell, R. (2018). Web Scraping with Python: Collecting More Data from the Modern Web. O'Reilly Media, Inc.
- Choudhary, A., Jain, S. (2021). Automation in Web Data Extraction: A Survey. Journal of Data Mining and Knowledge Discovery, 25(2), 305-328. Data, S. (2019). Techniques for Extracting Data from the Web: A Comparative Study. Journal of Web Engineering, 18(4), 389-412.

Chapter 4

EXISTING SOLUTION AND IT'S LIMITATIONS

4.1 EXISTING SYSTEM

Shilpa Chaudhari; R. Aparna; Vinay G Tekkur; G L. Pavan; Shreekanth R Karki [1] has used an algorithm to extract the information from the website about healthy diet. They have used different web scraping algorithms to detect the information from several websites and the best. They have used Python and MongoDB in this web scraping. This system helps to format the ingredients in the right way and in a healthy way.

Sumit Sakarkar; Vaibhav Chaudhari; Tanmay Gaurkar; Aditya Veer; and Mayura Kulkarni SCET have proposed a system that mainly recommends the information that the user is favorable. This is a scraping tool that extracts the data only based on the user's preference. This paper is mainly about evaluating user likeliness and user preferences and giving the data accordingly.

David Mathew Thomas; and Sandeep Mathur This paper is mainly about how data is extracted using Python software from different websites that have different algorithms to extract the data. In this paper, there is a specified language Python, which determines what is the methodology of the language and how it is implemented.

Amalia Amalia; Rizky Maulidya Afifa; Herriyance Herriyance [4] in this paper mainly focused on the tropical diseases information which is mainly faced by Indonesian people. The data extraction here is done on the diseases to know the medication they should use if they are infected by the diseases.

Ismael Camargo-Henríquez; Yarisel Núñez-Bernal[5] In this paper they have implemented data extracted based on social media such as Instagram and Facebook, which will extract the data that is trending online and help the people to be updated. This could extract the data which is popular enough and interesting.

Chun Feng Lin; and Sheng Chih Yang[6] have proposed a system that mainly focuses on Taiwan stock market prediction in which the web scraping mechanism is used which is time a time extractor because every minute the strokes may change their price. Through this, it is helpful to invest the money in the right stock.

Ayat Abodayeh; Reem Hejazi; Ward Najjar; Leena Shihadeh; and Rabia Latif[7] proposed a new algorithm which is Beautiful Soup which is a simple and easiest algorithm for extracting the data from the website. This form of extraction can extract whole data at a time.

4.2 LIMITATIONS OF EXISTING PROJECT

Web Scraping Challenges Explained in Detail:

- Learning Curve: While web scraping tools can be user-friendly, understanding fundamental web technologies like HTML and CSS is crucial for effective scraping. Navigating complex page structures, handling dynamic content, and dealing with anti-scraping measures may require coding knowledge and experience. Mastering advanced techniques like browser automation and API integration takes time and practice.
- Website Structure: Websites frequently update their layouts and content organization, making scraping scripts outdated. Complex structures with embedded elements, nested content, and dynamic loading require sophisticated scraping techniques. Adapting to frequent changes can be time-consuming and resource-intensive.
- Large Scale Data Extraction: Downloading large volumes of data can overload your computer and internet connection. Targeting websites with high traffic may trigger server-side throttling or blocks. Efficiently managing and storing vast amounts of scraped data requires proper planning and infrastructure.
- IP Bans: Websites can identify and block scraping attempts based on IP addresses. Using proxies or rotating IPs can mitigate this risk but adds complexity and cost. Persistent scraping from the same IP can lead to permanent bans, limiting future access.
- Legal Risks: Scraping without permission or violating website terms of service can be considered unlawful. Extracting large amounts of data can be misconstrued as a denial-of-service attack, leading to legal repercussions. Always ensure ethical and responsible scraping practices to avoid legal trouble.
- Data Availability: Content behind logins, paywalls, or requiring user interaction may be inaccessible to scrapers. Scraping incomplete or inaccurate data can lead to misleading results and inaccurate analysis. Consider alternative data sources or API access where available. 4

- Website Performance: Excessive scraping bots can overload website servers, causing slow loading times and reduced user experience. Ethical scraping involves responsible crawling frequencies and respecting website robots.txt guidelines.
- Website Analytics: Scraping bots can inflate website traffic data, distorting analytics and misrepresenting user behavior. Websites may implement measures to identify and exclude bot activity from their analytics. By understanding these challenges and approaching web scraping responsibly, you can avoid technical hurdles, legal issues, and ethical concerns, ensuring a smooth and successful data acquisition process.

Chapter 5

PROPOSED SYSTEM

The proposed approach is to develop a web application where the user can enter the link of a website they want from where the required data will be extracted. The validation of the link is done with the help of a Python library called Requests by sending a request to that page and with the help of another Python library called BeautifulSoup which generates a parse tree, the extraction of data is done. To store this extracted data and store the extracted data in that Excel sheet

5.1 OBJECTIVES OF PROPOSED SYSTEM

The objectives of the proposed system include the following:

- To extract required data from the unstructured website

5.2 SYSTEM REQUIREMENTS

Here are the requirements for developing and deploying the application.

5.2.1 SOFTWARE REQUIREMENTS

Below are the software requirements for the application development:

- The required languages are Python, HTML, CSS.
- Editor for Python - PyCharm or VSCode
- Python libraries - Flask,BeautifulSoup, Requests, Pandas.
- Google Chrome, Firefox, Microsoft Edge, or Brave Browser

SOFTWARE REQUIREMENTS	
LANGUAGES	LIBRARIES
PYTHON	FLASK
HTML	BEAUTIFULSOUP
CSS	REQUESTS
	PANDAS

Table 5.1: SOFTWARE REQUIREMENTS

5.2.2 HARDWARE REQUIREMENTS

Below are the hardware requirements for the application development:

- Operating System: Windows
- Processor: intel i3(min)
- Ram: 4 GB(min)
- Hard Disk: 250GB(min)

HARDWARE REQUIREMENTS
OPERATING SYSTEM(WINDOWS)
PROCESSOR(Intel i3 Minimum)
RAM(4GB Minimum)
HARDISK(256GB Minimum)

Table 5.2: HARDWARE REQUIREMENTS

5.2.3 FUNCTIONAL REQUIREMENTS

- The user interface of the software should be able to take a link, html tag, and excel sheet file name as input from the user and then the link is validated and requested, then extracting the required data.
- The extracted data should be stored in the required Excel sheet.
- After the extraction and storage, the Excel file should be visible on the user's computer with the required data.

5.2.4 NON-FUNCTIONAL REQUIREMENTS

Reliability

- Regardless of the number of attempts the system should be able to validate the link and extract the required data.
- The system should be able to handle any exception properly.
- As for the output, the system should be able to provide a faster response.

Scalability

- To produce better results, the system should be able to validate any website and extract.
- The system must be able to cope up with any kind of website

5.3 TECHNOLOGIES USED IN THE PROPOSED SYSTEM

PYTHON

Python stands out as a dynamic, high-level programming language that prioritizes both readability and rapid development. It embraces object-oriented structures, flexible data handling, and seamless integration with existing components. Python's modular nature promotes code reusability, making it a versatile tool for diverse projects. As an open-source language, it's freely accessible and adaptable across major platforms, empowering developers with its extensive standard library and interpreter.

PYTHON LIBRARIES

- REQUESTS

Requests module allows you to send HTTP requests using Python. The HTTP request returns a Response Object with all the response data (content, encoding, status, etc)

- BEAUTIFULSOUP

BeautifulSoup is a library that makes it easy to scrape information from web pages which generates a parse tree. It sits atop an HTML or XML parser, providing Pythonic idioms for iterating, searching, and modifying the parse tree.

- PANDAS

Pandas is a powerful and popular open-source library in Python specifically designed for data manipulation and analysis.

- HTML

HTML is the code behind every webpage, it is like the skeleton and muscles of a website. It uses tags to define content - headings, paragraphs, images, links - but lacks the visual flair. That's where CSS comes in, applying the paint and decorations. Simple to learn, HTML is the essential alphabet of web development, the foundation for every website, paving the way for further creation and exploration in the vast digital landscape.

- FLASK

Flask is a lightweight web framework for Python that is designed to make it easy to get started with web development. It provides the essentials for building web applications, including routing, templates, and request handling, without imposing a particular structure or dependencies. Flask is highly flexible, allowing developers to use extensions and customize components as needed.

5.4 DATA SET USED IN THE PROPOSED SYSTEM

The required data is collected from the following wikipedia ,the provided link consists of many paragraphs,links, and headings. The link is validated and access is requested, after granting permission extraction using BeautifulSoup is done on the link to extract required data from the website and is stored in an Excel sheet

Chapter 6

SYSTEM DESIGN

6.1 COMPONENTS OR USERS IN THE PROPOSED SYSTEM

Requester

Requests module allows you to send HTTP requests using Python. The HTTP request returns a Response Object with all the response data (content, encoding, status, etc)

Extractor

After getting access to the website, BeautifulSoup starts extracting the required data from the webpage, and then using the Pandas library, the extracted data is stored in a CSV file.

End user

The end user is the person who wants the requested data. The end user can access the extracted data by opening the Excel sheet created by the extractor. After the extraction, the final result is stored in the user's device.

6.2 PROPOSED SYSTEM ARCHITECTURE

An architectural diagram outlines the system's components, their relationships, and system functionality. A link to the target web page is given, then it is requested for granting access then extracting the required data takes place. Users input the target web page link, target tag, and file name for the Excel sheet to store the extracted data. The final output is then stored in the user's device.

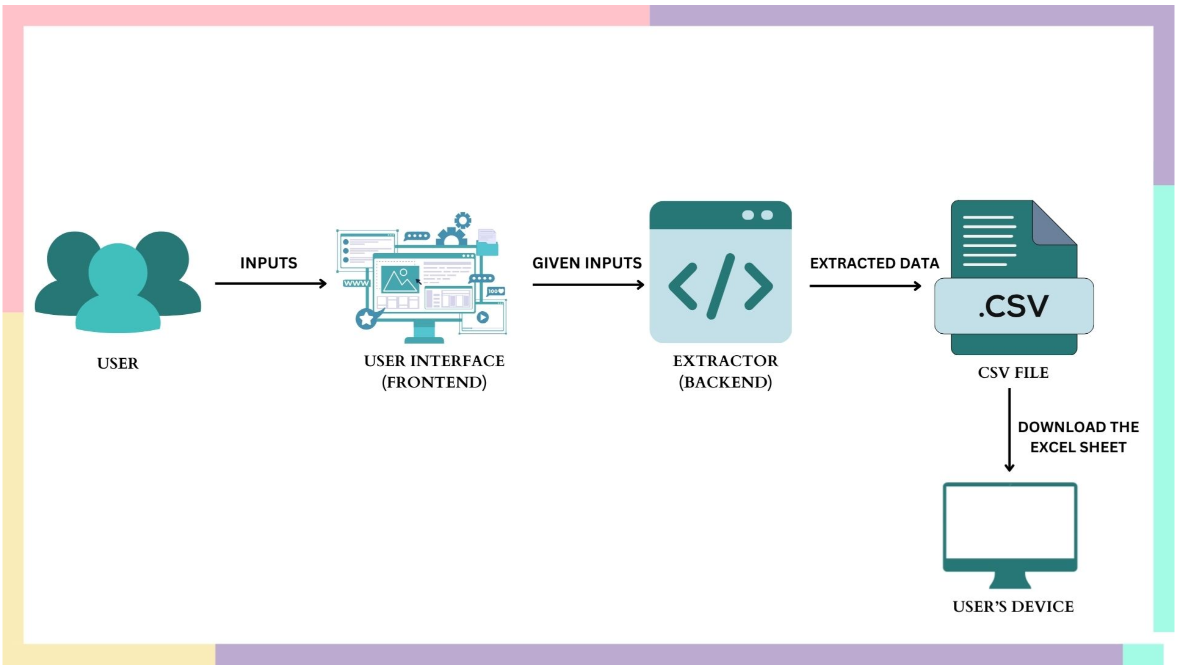


Figure 6.1: SYSTEM ARCHITECTURE

6.3 UML DIAGRAMS

6.3.1 USE CASE DIAGRAM

The use case diagram describes the function and scope of the system and also the system requirements. Our use case diagram has two actors - the user and the backend. Each actor interacts with a particular use case. The user actor interacts with the interface to input the required data and obtain the final result from the backend.

The backend is the main python code which takes in the inputs given by the user as parameters and request the target web page and starts extracting once the request is granted, after extraction the extracted data is stored in a csv file which is downloded in users computer.

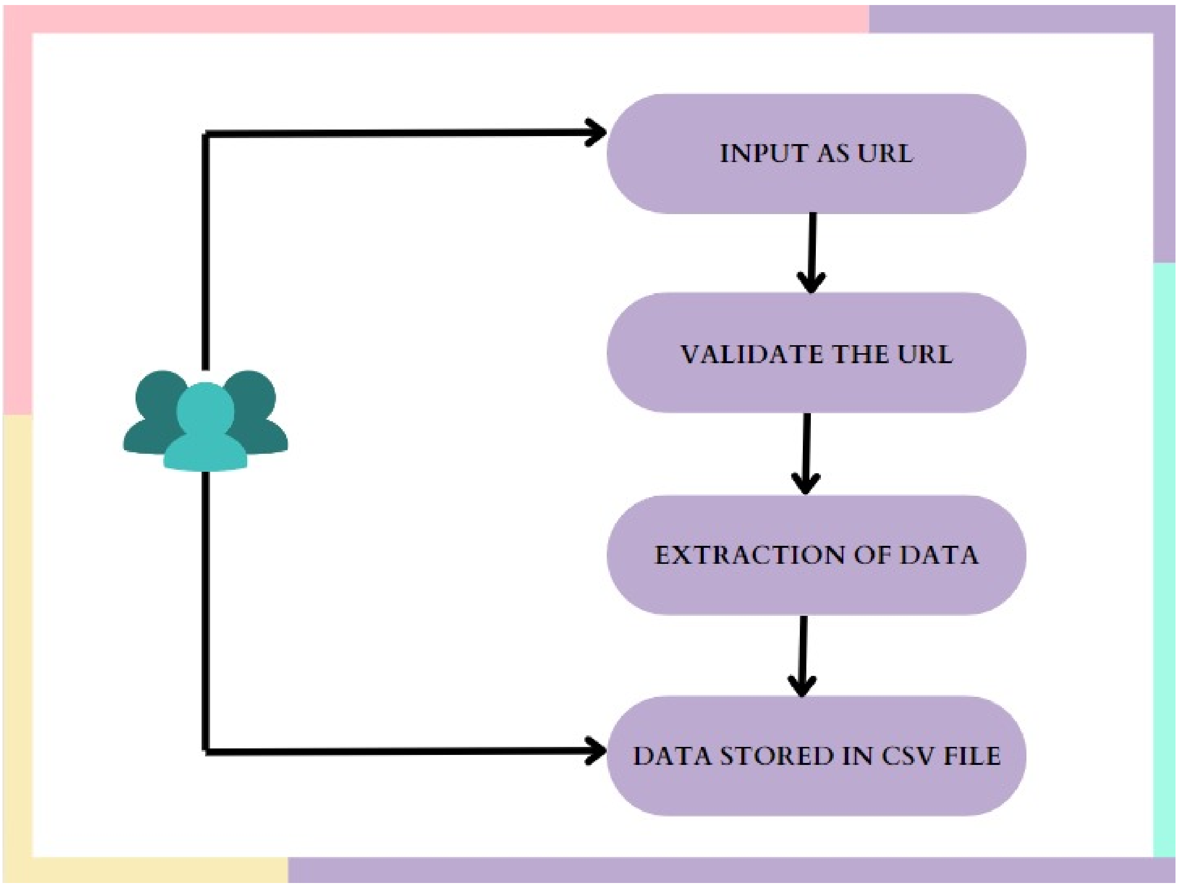


Figure 6.2: USE CASE DIAGRAM

6.3.2 SEQUENCE DIAGRAM

The sequence diagram depicts the processes involved and the sequence of messages exchanged between the processes needed to carry out the functionalities.

The sequence of the application is as follows:

1. Client (Web Scraper): The entity initiating the web scraping process. It can be a script, program, or tool specifically designed for this task.
2. HTTP Request: The client sends an HTTP request to the target website's server, specifying the desired web page or resourc
3. Website Server: The server receives the request, processes it, and locates the requested content.
4. HTTP Response: The server sends an HTTP response back to the client, containing the requested content (HTML, CSS, JavaScript, etc.).
5. Content Parsing: The client receives the response and parses it using parsing libraries or tools.

6. Data Extraction: The client extracts the specific data elements of interest from the parsed content, often using techniques like:

   - DOM traversal (navigating the HTML structure)
   - Regular expressions (matching patterns in text)
   - CSS selectors (targeting specific elements)

7. Data Storage: The extracted data is stored in a suitable format, such as a database, CSV file, or JSON file.

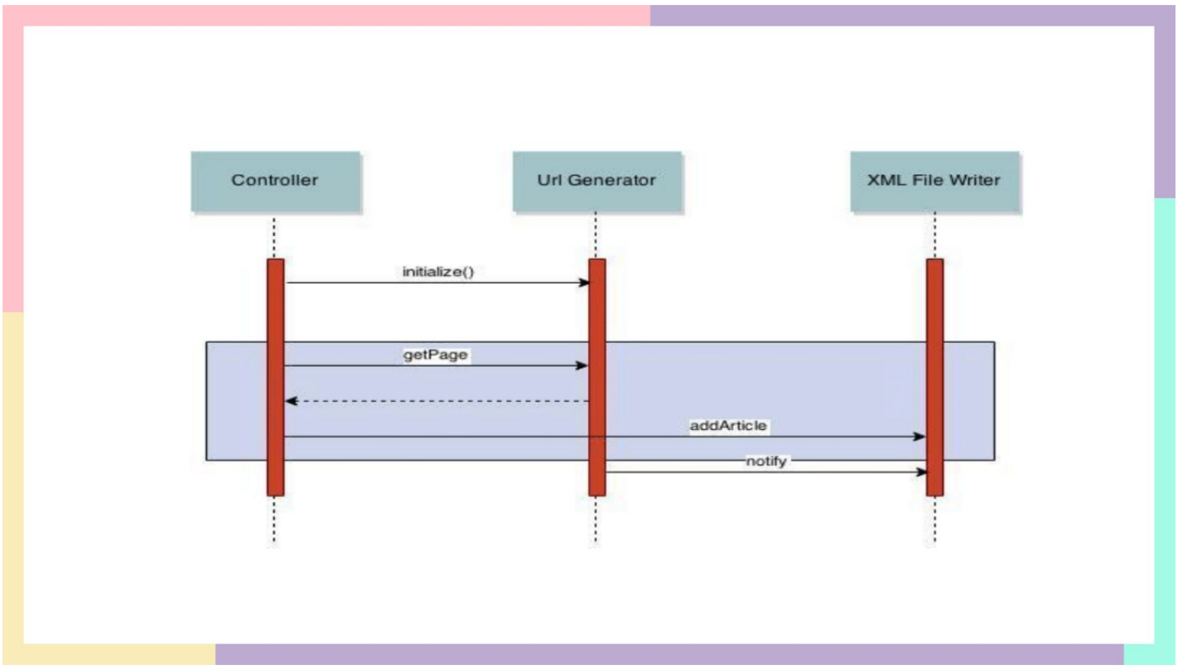


Figure 6.3: SEQUENCE DIAGRAM

6.3.3 ACTIVITY DIAGRAM

The activity diagram provides a view of the behavior of the application by the sequence of the actions in a process.

Activity Diagram

- Extractor

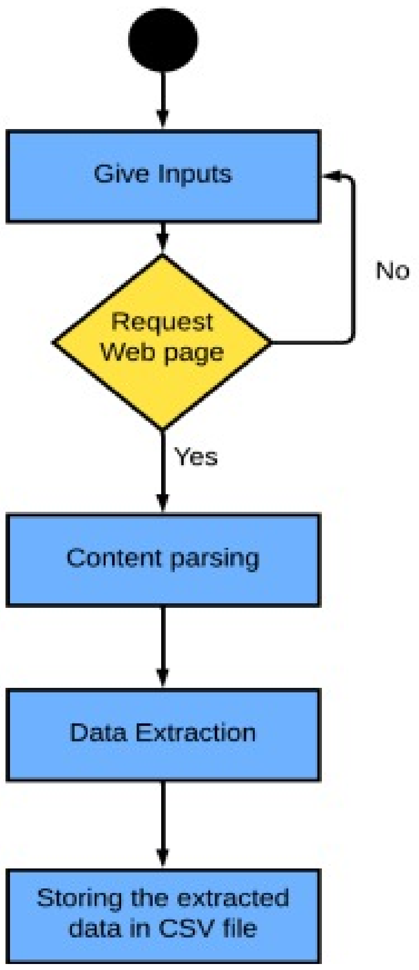


Figure 6.4: ACTIVITY DIAGRAM

1. The initial state of the project includes providing the requested data details as input by the user to the developed extension.
2. The inputs are passed to the backend. The backend is developed using python and javascript.
3. The backend sends a HTTPS request and after getting access data extraction starts
4. The final output is an excel sheet in the user's device which consists of the required data from the webpage
    - If the targeted website doesn't give access or can't find the required data then there will be no excel sheet in the user's device.

6.4 MODULE DIAGRAM

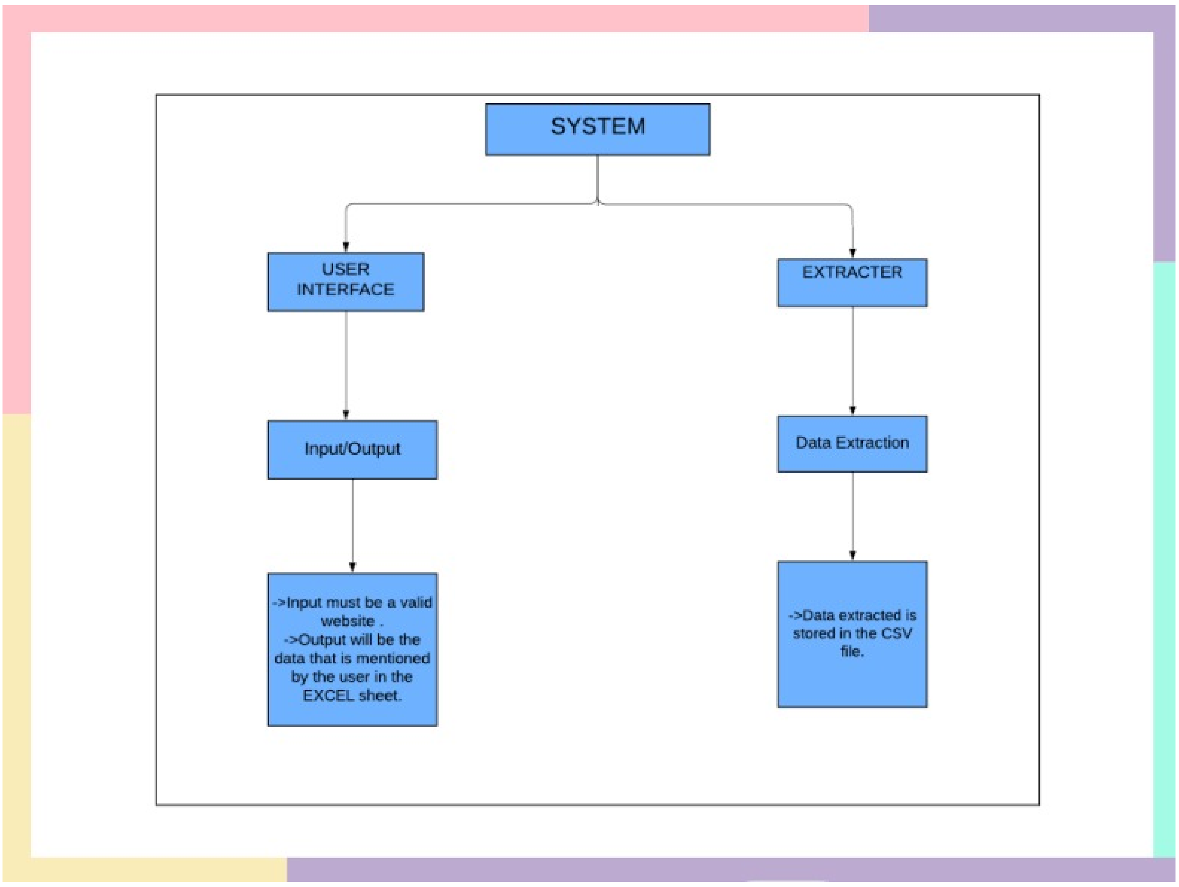

SYSTEM
USER INTERFACE
EXTRACTER
Input/Output
Data Extraction
->Input must be a valid website .
->Output will be the data that is mentioned by the user in the EXCEL sheet.
->Data extracted is stored in the CSV file.

Figure 6.5: MODULE DIAGRAM

The proposed system consists of three modules as mentioned in the above diagram.

- User Interface
- Extractor

USER INTERFACE

The functions in the user interface module include users giving the required inputs. Once the user gives the required input, the inputs are passed to the extractor to begin the extraction. This module allows the users to give input and and then as output the excel file containing the extracted data is downloaded into the user's device.

EXTRACTOR

The Firebase is used to store the csv file temporarily which contains the required data and downloads that csv file as an excel sheet into the user's device.

Chapter 7

IMPLEMENTATION OF PROJECT

7.1 Modules description

This project involves developing a web scraping tool using Python, Flask, HTML, and CSS. The tool will extract data from websites based on specified HTML tags, classes, and links. The extracted data will be stored in an Excel sheet and will be downloadable by the user.

MODULES
WEB SCRAPING MODULE
BACKEND SERVER MODULE
FRONTEND INTERFACE MODULE
CSV EXPORT MODULE

Table 7.1: MODULE DESCRIPTION

7.1.1 WEB SCRAPING MODULE

- Technology: Python (BeautifulSoup, requests)
- Functionality: - Accept user input for HTML tag, class, and website link.

7.1.2 BACKEND SERVER MODULE

- Technology: Flask
- Functionality: Create an API to accept scraping parameters from the front-end.
- Process the scraping request and return the extracted data.
- Manage user sessions and handle multiple requests.

PYTHON SOURCE CODE : app.py

```
from flask import Flask, request, render_template, send_file
import pandas as pd
import requests
from bs4 import BeautifulSoup

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/scrape', methods=['POST'])
def scrape():
    url = request.form['url']
    tag = request.form['tag']
    class_name = request.form['class_name']
    file_name = request.form['file_name']

    response = requests.get(url)
    soup = BeautifulSoup(response.content, 'html.parser')

    if class_name:
        data = soup.find_all(tag, class_=class_name)
    else:
        data = soup.find_all(tag)

    extracted_data = [item.get_text(strip=True) for item in data]

    df = pd.DataFrame(extracted_data, columns=['Data'])
    csv_path = f"{file_name}.csv"
    df.to_csv(csv_path, index=False)
    return send_file(csv_path, as_attachment=True)

if __name__ == '__main__':
    app.run(host='127.0.0.1', port=5000, debug=True)
```

CODE SUMMARY

This Flask application allows users to scrape text data from a webpage by specifying a URL, HTML tag, and optional CSS class. After submission, it extracts text content based on these parameters, stores it in a CSV file named by the user, and then lets them download this file. The application integrates Flask for web serving, Pandas for data handling, requests for fetching webpage content, and BeautifulSoup for parsing HTML. It's a simple yet effective tool for basic web scraping tasks with a user-friendly interface.

7.1.3 FRONTEND INTERFACE MODULE

- Technology: HTML, CSS
- Functionality: Design a user-friendly interface to accept user inputs for scraping.

FRONTEND SOURCE CODE

- **index.html**

```
    <!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Web Scraper</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <h1>WEB SCRAPING FROM WEBSITES</h1>
        <form action="/scrape" method="post">
            <label for="url">Website URL:</label>
            <input type="text" id="url" name="url" required>
            <label for="tag">HTML Tag:</label>
            <input type="text" id="tag" name="tag" required>
            <label for="class_name">Class Name (optional):</label>
            <input type="text" id="class_name" name="class_name">
            <label for="file_name">CSV File Name:</label>
            <input type="text" id="file_name" name="file_name" required>
            <input type="submit" value="SCRAPE">
        </form>
    </div>
    <div class="bottom-right-text">
        K.VENKATA ADITHYA - 22R21A6792 (Team Lead)
        <br>
        K.TAUFEEQ UMAR - 22R21A6786
        <br>
        B.VARUN - 22R21A6768
    </div>
</body>
</html>
```

CODE SUMMARY

This HTML code creates a user interface where users can input a website URL, specify an HTML tag to scrape, optionally provide a CSS class name, and define a name for the resulting CSV file. Upon submitting the form, this data is sent to a Flask backend for web scraping, processing, and CSV file generation. The page also includes static content displaying the names and IDs of team members involved in the project.

- **style.css**

```
body {
    font-family: Arial, sans-serif;
    background-color: #80CBC4;
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}

.container {
    background-color: #546E7A;
    padding: 2em;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    max-width: 500px;
    width: 100%;
    margin-bottom: 1em;
    border: 3px solid #000000;

}

h1 {
    margin-bottom: 1em;
    font-size: 2em;
    color: #00BCD4;
    text-align: center;
```

```
}

form {
    display: flex;
    flex-direction: column;
}

label {
    margin-bottom: 0.5em;
    color: #ffffff;
}

input[type="text"] {
    padding: 0.5em;
    margin-bottom: 1em;
    border: 3px solid #000000;
    border-radius: 4px;
    font-size: 1em;
}

input[type="submit"] {
    padding: 0.5em;
    background-color: #00BCD4;
    color: #0e5345 ;
    font-weight: bold;
    border-radius: 15px;
    font-size: 1em;
    cursor: pointer;
    width: 100px;
    margin-left: 200px;
    height: 45px;


}

input[type="submit"]:hover {
    background-color: #179f84;
}
.bottom-right-text {
    position: absolute;
    bottom: 20px;
    right: 20px;
    background-color: #117864 ;
    color: #fff;
    padding: 10px;
    border-radius: 10px;
    font-size: 0.9em;
    margin-bottom: 1em;
    border: 3px solid #000000;

}
```

CODE SUMMARY

The CSS stylesheet enhances the web scraping application's interface with a modern, readable design. It utilizes a consistent font style, a calming background color, and well-defined containers for content. Key elements such as headings, form inputs, and buttons are styled for clarity and usability, while additional information is presented neatly at the bottom-right corner. Overall, the design balances aesthetics with functionality, ensuring a pleasant user experience across different devices and screen sizes.

7.1.4 CSV EXPORT MODULE

- Technology:Python (pandas)
- Functionality: Convert the extracted data into an Excel sheet.
- Provide a download link for the generated Excel file.

EXPORT MODULE SOURCE CODE:

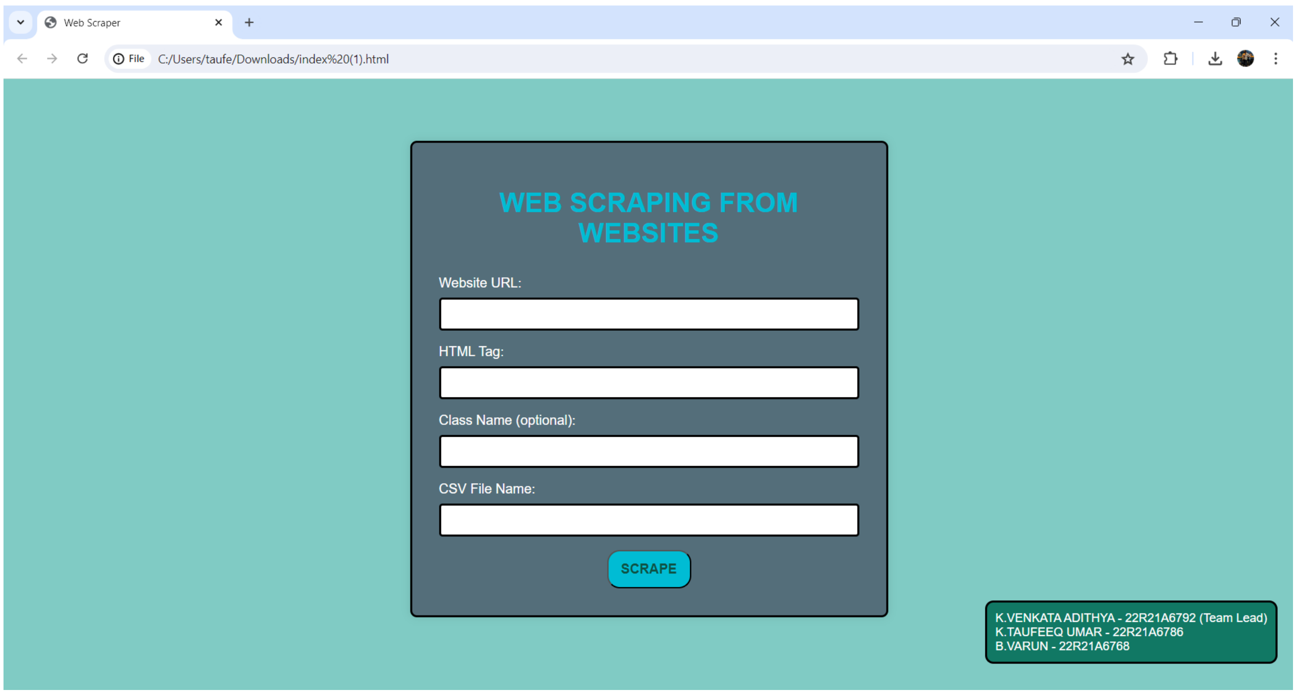
```
    # webscraper.py
from bs4 import BeautifulSoup
import requests
import pandas as pd

def scrape_website(url, tag, class_name=None):
    """
    Scrapes data from a webpage based on provided tag and optional class_name.

    Args:
    - url (str): URL of the webpage to scrape.
    - tag (str): HTML tag to search for.
    - class_name (str, optional): CSS class name to filter within the tag.

    Returns:
    - list: List of extracted text data.
    """
    response = requests.get(url)
    soup = BeautifulSoup(response.content, 'html.parser')

    if class_name:
        data = soup.find_all(tag, class_=class_name)
    else:
        data = soup.find_all(tag)

    extracted_data = [item.get_text(strip=True) for item in data]
    return extracted_data
```

```
def save_to_csv(data, file_name):
    """
    Saves extracted data to a CSV file.

    Args:
    - data (list): List of data to save.
    - file_name (str): Name of the CSV file to create.

    Returns:
    - str: Path to the saved CSV file.
    """
    df = pd.DataFrame(data, columns=['Data'])
    csv_path = f"{file_name}.csv"
    df.to_csv(csv_path, index=False)
    return csv_path
```

Chapter 8

RESULTS

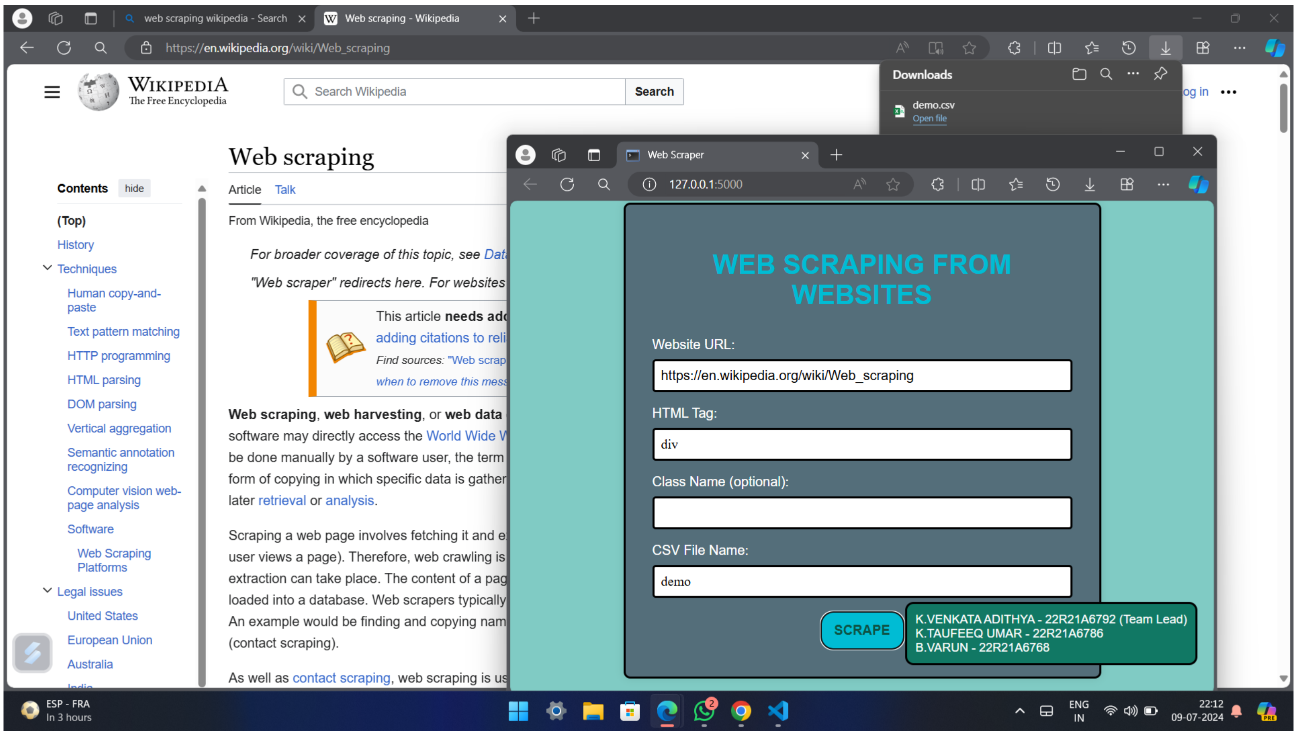
8.0.1 INTERFACE SCREENSHOT



K.VENKATA ADITHYA - 22R21A6792 (Team Lead)
K.TAUFEEQ UMAR - 22R21A6786
B.VARUN - 22R21A6768

Figure 8.1: INTERFACE SCREENSHOT

8.0.2 OUTPUT SCREENSHOTS



WEB SCRAPING FROM WEBSITES

Website URL: https://en.wikipedia.org/wiki/Web_scraping

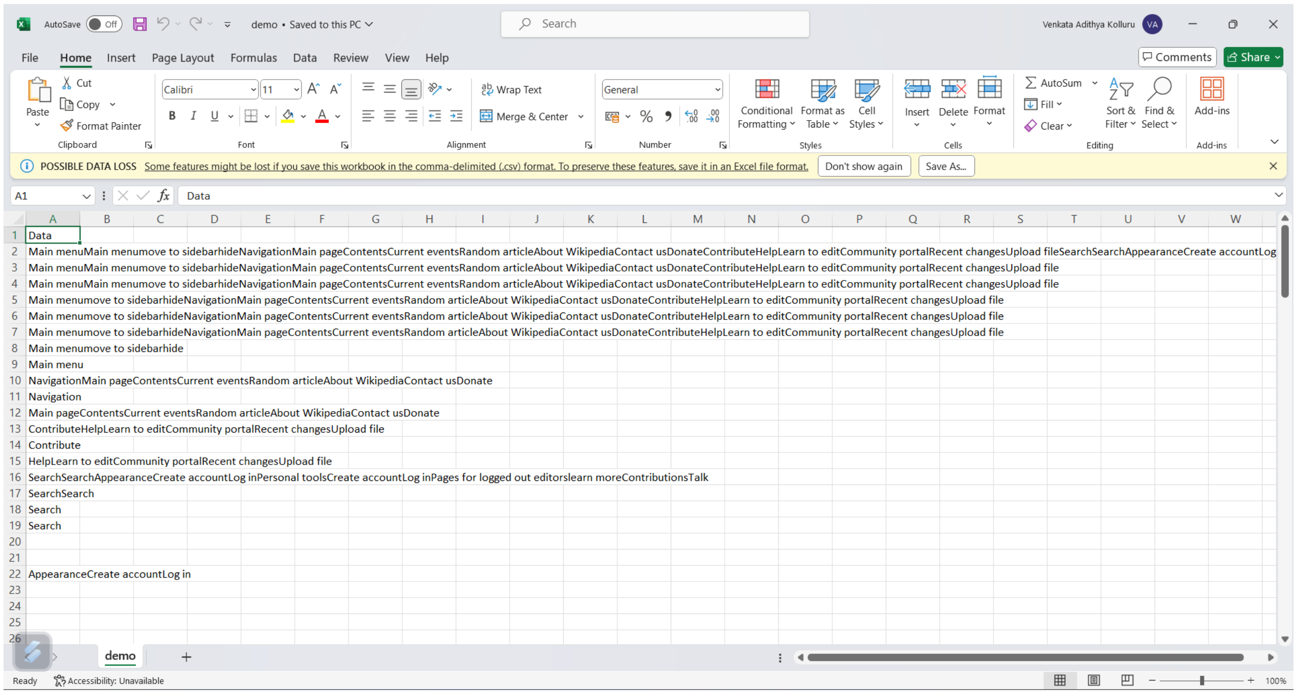
HTML Tag: div

Class Name (optional):

CSV File Name: demo

K.VENKATA ADITHYA - 22R21A6792 (Team Lead)
K.TAUFEEQ UMAR - 22R21A6786
B.VARUN - 22R21A6768

Figure 8.2: INTERFACE SCREENSHOT 2



POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format.

	A
1	Data
2	Main menuMain menumove to sidebarhideNavigationMain pageContentsCurrent eventsRandom articleAbout WikipediaContact usDonateContributeHelpLearn to editCommunity portalRecent changesUpload fileSearchSearchAppearanceCreate accountLog
3	Main menuMain menumove to sidebarhideNavigationMain pageContentsCurrent eventsRandom articleAbout WikipediaContact usDonateContributeHelpLearn to editCommunity portalRecent changesUpload file
4	Main menuMain menumove to sidebarhideNavigationMain pageContentsCurrent eventsRandom articleAbout WikipediaContact usDonateContributeHelpLearn to editCommunity portalRecent changesUpload file
5	Main menumove to sidebarhideNavigationMain pageContentsCurrent eventsRandom articleAbout WikipediaContact usDonateContributeHelpLearn to editCommunity portalRecent changesUpload file
6	Main menumove to sidebarhideNavigationMain pageContentsCurrent eventsRandom articleAbout WikipediaContact usDonateContributeHelpLearn to editCommunity portalRecent changesUpload file
7	Main menumove to sidebarhideNavigationMain pageContentsCurrent eventsRandom articleAbout WikipediaContact usDonateContributeHelpLearn to editCommunity portalRecent changesUpload file
8	Main menumove to sidebarhide
9	Main menu
10	NavigationMain pageContentsCurrent eventsRandom articleAbout WikipediaContact usDonate
11	Navigation
12	Main pageContentsCurrent eventsRandom articleAbout WikipediaContact usDonate
13	ContributeHelpLearn to editCommunity portalRecent changesUpload file
14	Contribute
15	HelpLearn to editCommunity portalRecent changesUpload file
16	SearchSearchAppearanceCreate accountLog inPersonal toolsCreate accountLog inPages for logged out editorslearn moreContributionsTalk
17	SearchSearch
18	Search
19	Search
20	
21	
22	AppearanceCreate accountLog in
23	
24	
25	
26	

Figure 8.3: OUTPUT SCREENSHOT

Chapter 9

CONCLUSION

Web data extraction as a tool for information gathering, focusing on ethical practices and accessibility. By using the right methods and staying small-scale, we've shown how this technique can be effective for personal use and research without breaking the bank. By following legal and ethical guidelines, we've promoted responsible data collection and use, ensuring these valuable online resources stay available for everyone. Web data extraction will undoubtedly play a more and more important role in various professional and academic fields.

9.1 REFERENCE

[1]. Ingredient/Recipe Algorithm using Web Mining and Web Scraping for Smart Chef
https://ieeexplore.ieee.org/document/9198450/keywordskeywords
Shilpa Chaudhari; R. Aparna; Vinay G Tekkur; G L. Pavan; Shreekanth R Karki.
DOI:10.1109/CONECCT50063.2020.9198450

[2]. Web Personalisation based on User Interaction: Web Personalisation
https://ieeexplore.ieee.org/document/9388384
Sumit Sakarkar; Vaibhav Chaudhari; Tanmay Gaurkar; Aditya Veer; Mayura Kulka SCET
DOI:10.1109/ICICV50876.2021.9388384

[3]. Data Analysis by Web Scraping using Python
https://ieeexplore.ieee.org/document/8822022
David Mathew Thomas; Sandeep Mathur
DOI: 10.1109/ICECA.2019.8822022

[4]. Resource Description Framework Generation for Tropical Disease Using Web Scraping

https://ieeexplore.ieee.org/document/8684030
Amalia Amalia; Rizky Maulidya Afifa; Herriyance Herriyance
DOI: 10.1109/COMNETSAT.2018.8684030

[5]. A Web Scraping based approach for data research through social media: An Instagram case
https://ieeexplore.ieee.org/document/9941290
Ismael Camargo-Henríquez; Yarisel Núñez-Bernal
DOI: 10.1109/AmITIC55733.2022.9941290

[6]. Web Scraping to Implement Tape Reading on Taiwan Stock Periodically with GUI
https://ieeexplore.ieee.org/document/9645633 Chun Feng Lin; Sheng Chih Yang
DOI: 10.1109/ECICE52819.2021.9645633

[7]. Web Scraping for Data Analytics: A BeautifulSoup Implementation
https://ieeexplore.ieee.org/document/10145369
Ayat Abodayeh; Reem Hejazi; Ward Najjar; Leena Shihadeh; Rabia Latif
DOI: 10.1109/WiDS-PSU57071.2023.00025

[8]. https://oxylabs.io/blog/python-web-scraping

[9]. https://en.wikipedia.org/wiki/Webscraping

[10]. https://uiverse.io/