

LABORATORY REPORT
Application Development Lab
(CS33002)

B.Tech Program in CSE

Submitted By

Name:- Rishi Banerjee

Roll No: 2205151



Kalinga Institute of Industrial Technology
(Deemed to be University)
Bhubaneswar, India

Spring 2024-2025

Table of Content

Exp No.	Title	Date of Experiment	Date of Submission	Remarks
1.	Build a Resume using HTML/CSS	16/01/2025	23/01/2025	
2.	Machine Learning for Cat and Dog Classification	23/01/2025	30/01/2025	
3.	Pneumonia Detection using CNN			
4.	Regression Analysis for Stock Prediction	30/01/2025	06/02/2025	
5.	Conversational Chatbot with Any Files	06/02/2025	20/02/2025	
6.	Web Scraper using LLMs	13/03/2025	20/03/2025	
7.				
8.				
9.	Open Ended 1			
10.	Open Ended 2			

Lab Number	8
Experiment Number	6
Experiment Title	Web Scraper using LLMs
Date of Experiment	13/03/2025
Date of Submission	20/03/2025

1. Objective:-

To create a web scraper application integrated with LLMs for processing scraped data.

2. Procedure:- (Steps Followed)

1. Use Python libraries like BeautifulSoup and Requests to scrape web data.
2. You can also use LlamaIndex for Web Scraping and Ollama for open ended LLMs.
3. Integrate LLMs to process and summarize the scraped information.
4. Develop a Flask backend for handling scraping tasks and queries.
5. Create an HTML/CSS frontend to initiate scraping (like the web page to scrape) and display results.
6. You can also take a topic and search the web for a web page and then scrape it.

3. Code:-

```

app.py
backend > app.py
1 from flask import Flask, request, jsonify
2 import re
3 import requests
4 from bs4 import BeautifulSoup
5 from youtube_transcript_api import YouTubeTranscriptApi
6 from duckduckgo_search import DDGS # Corrected import
7 import google.generativeai as genai # Using Gemini API for LLM-based text processing
8 from dotenv import load_dotenv
9 import os
10 from flask_cors import CORS
11
12 # Initialize Flask app
13 app = Flask(__name__)
14 CORS(app) # Allow all origins
15 load_dotenv() # Load environment variables from .env
16
17 # Configure Gemini API
18 genai.configure(api_key=os.getenv("GEMINI_API_KEY"))
19
20 # Function to classify user input
21 def classify_input(user_text):
22     url_pattern = re.compile(r'https?://\S+')
23     youtube_pattern = re.compile(r'(https://)?(www\.)?(youtube\.com|youtu\.?be)/.+')
24
25     urls = url_pattern.findall(user_text)
26     youtube_links = [url for url in urls if youtube_pattern.match(url)]
27
28     if youtube_links:

```

```
app.py x
backend > app.py
1 from flask import Flask, request, jsonify
2 import re
3 import requests
4 from bs4 import BeautifulSoup
5 from youtube_transcript_api import YouTubeTranscriptApi
6 from duckduckgo_search import DDGS # Corrected import
7 import google.generativeai as genai # Using Gemini API for LLM-based text processing
8 from dotenv import load_dotenv
9 import os
10 from flask_cors import CORS
11
12 # Initialize Flask app
13 app = Flask(__name__)
14 CORS(app) # Allow all origins
15 load_dotenv() # Load environment variables from .env
16
17 # Configure Gemini API
18 genai.configure(api_key=os.getenv("GEMINI_API_KEY"))
19
20 # Function to classify user input
21 def classify_input(user_text):
22     url_pattern = re.compile(r'https?://\S+')
23     youtube_pattern = re.compile(r'(https://)?(www\.)?(youtube\.com|youtu\.?be)/.+')
24
25     urls = url_pattern.findall(user_text)
26     youtube_links = [url for url in urls if youtube_pattern.match(url)]
27
28     if youtube_links:
```

```
pagetx x
frontend > src > app > pagetx > Home
1 "use client";
2 import { useState } from "react";
3 import axios from "axios";
4 import ReactMarkdown from "react-markdown";
5 import { motion } from "framer-motion";
6
7 Qodo Gen: Options | Test this function
8 export default function Home() {
9     const [inputText, setInputText] = useState("");
10     const [responseText, setResponseText] = useState("");
11     const [loading, setLoading] = useState(false);
12     const [showResponse, setShowResponse] = useState(false);
13
14     const handleSubmit = async () => {
15         if (!inputText) return;
16
17         setLoading(true);
18         setShowResponse(false); // Hide response initially
19         try {
20             const response = await axios.post(
21                 `${process.env.NEXT_PUBLIC_BACKEND_URL}/process`,
22                 { text: inputText }
23             );
24             setResponseText(response.data.summary);
25             setShowResponse(true); // Show response after fetching
26         } catch (error) {
27             setResponseText("Error fetching data.");
28             setShowResponse(true);
29         }
30         setLoading(false);
31     };
32
33     return (
34         <div className="flex flex-col items-start w-full justify-center min-h-screen bg-gray-900 text-white relative">
35             <div className="flex flex-col items-center justify-center p-6 transition-transform duration-300 ease-in-out ${showResponse ? "w-[50%]" : "w-full"}">
36                 <motion.h1
37                     className="text-4xl font-bold mb-6 text-center text-blue-500 drop-shadow-custom no-select flex items-center justify-center"
38                     initial={{ opacity: 0, y: -20 }}
39                     animate={{ opacity: 1, y: 0 }}
40                     transition={{ duration: 0.5 }}
41                 >
42                      WebScrap AI
43                 </motion.h1>
44                 <motion.p
45                     className="text-md mb-6 text-center text-gray-500 drop-shadow-custom no-select"
46                     initial={{ opacity: 0, y: -20 }}
47                     animate={{ opacity: 1, y: 0 }}
48                     transition={{ duration: 0.5 }}
49                 >
50                     Scrape websites & YouTube videos effortlessly. <br/> Extract key insights, summaries, and data in seconds.
51                 </motion.p>
52                 <motion.textarea
53                     className="w-full shadow-[inset 4px 4px 8px rgba(0,0,0,0.25)] p-3 rounded-lg border max-w-lg border-gray-700 bg-gray-800
54                     focus-within:ring-2 focus-within:ring-blue-500 focus-within:border-blue-500 transition-all duration-300 no-select"
55                     placeholder="Enter text, URL, or keywords..."
56                     value={inputText}
57                     onChange={(e) => setInputText(e.target.value)}
58                     rows={4}
59                     initial={{ opacity: 0, scale: 0.9 }}
60                     animate={{ opacity: 1, scale: 1 }}
61                     transition={{ duration: 0.4 }}
62                 >
```

4. Results/Output:-

The screenshot displays the WebScrap AI web application. The top section features the logo and a brief description: 'Scrape websites & YouTube videos effortlessly. Extract key insights, summaries, and data in seconds.' Below this is an input field containing the text 'Pokemon news' and a 'Submit' button. The bottom section is divided into two columns. The left column shows the input field again, and the right column displays the 'Response:' which includes a detailed explanation of why a website might be flagged as suspicious, listing causes like disabled cookies, JavaScript, browser extensions, VPN/proxy usage, rapid requests, and outdated browsers. It also provides specific solutions for each cause, such as enabling cookies in Chrome, Firefox, Edge, and Safari, and enabling JavaScript in Chrome and Firefox. The footer of the interface includes a 'QwertyFusion' logo and a 'GitHub Repository' link.

WebScrap AI
Scrape websites & YouTube videos effortlessly
Extract key insights, summaries, and data in seconds

Enter text, URL, or keywords...

Submit

Response:
This message means the website you're trying to access thinks your browser activity is suspicious and resembles a bot's behavior. Here's a breakdown of the possible causes and solutions.

Possible Causes:
Cookies Disabled: Cookies are small files websites use to track your activity and preferences. If disabled, the site can't remember you, making you look like a new, potentially malicious user with each request.
JavaScript Disabled: JavaScript is a programming language that makes websites dynamic and interactive. Many websites rely on it for essential functions. Disabling it can break a site's functionality and make your browser look unusual.
Browser Extensions: Some browser extensions, especially those related to privacy, ad blocking, or security, can sometimes interfere with website functionality or be flagged as suspicious.
VPN/Proxy Usage: While VPNs are generally safe, some websites are sensitive to VPNs or proxies, especially those used for malicious purposes. Your IP address might be associated with bot activity.
Rapid Requests/Automated Behavior: If you're clicking links or refreshing the page too quickly, it might be interpreted as automated behavior.
Outdated Browser: An old browser might not support the latest web standards, making it look incompatible and potentially suspicious.
Privacy-Focused Browsers/Settings: Browsers like Brave or Tor, or overly strict privacy settings, can make your browser fingerprint unique, sometimes triggering bot detection.
Network Issues: Less likely, but intermittent network issues could lead to incomplete requests, causing the website to flag you.

Solutions:
Enable Cookies:
Chrome: Go to `chrome://settings/content/cookies`. Make sure "Allow all cookies" is selected or that the specific website is allowed.
Firefox: Go to `about:preferences#privacy`. Under "Cookies and Site Data," select "Standard" or "Custom" and ensure "Cookies" are enabled. If you use "Custom," make sure to not block "Third-Party Cookies".
Edge: Go to `edge://settings/content/cookies`. Make sure "Allow sites to save and read cookie data (recommended)" is enabled.
Safari: Go to Safari > Preferences > Privacy. Ensure "Prevent cross-site tracking" is *unchecked* and "Block all cookies" is *unchecked*.
Enable JavaScript:
Chrome: Go to `chrome://settings/content/javascript`. Make sure "Sites can use JavaScript" is selected.
Firefox: Go to `about:preferences#privacy`. JavaScript cannot be directly disabled here. Disabling script is controlled by browser add-ons (NoScript).
Extra: Go to `chrome://settings/content/permissions`. Make sure "Allow all permissions" is selected.

5. Remarks:-

Signature of the Student

(Name of the Student)

Signature of the Lab Coordinator

(Name of the Coordinator)