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

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

```python
from flask import Flask, request, jsonify
import re
import requests
from bs4 import BeautifulSoup
from youtube_transcript_api import YouTubeTranscriptApi
from duckduckgo_search import DDGS  # Corrected import
import google.generativeai as genai  # Using Gemini API for LLM-based text processing
from dotenv import load_dotenv
import os
from flask_cors import CORS

# Initialize Flask app
app = Flask(__name__)
CORS(app)  # Allow all origins
load_dotenv()  # Load environment variables from .env

# Configure Gemini API
genai.configure(api_key=os.getenv("GEMINI_API_KEY"))

# Function to classify user input
def classify_input(user_text):
    url_pattern = re.compile(r'https?://\S+')
    youtube_pattern = re.compile(r'(https?://)?(www\.)?(youtube\.com|youtu\.?be)/.+')

    urls = url_pattern.findall(user_text)
    youtube_links = [url for url in urls if youtube_pattern.match(url)]

    if youtube_links:
```

app.py

backend > app.py

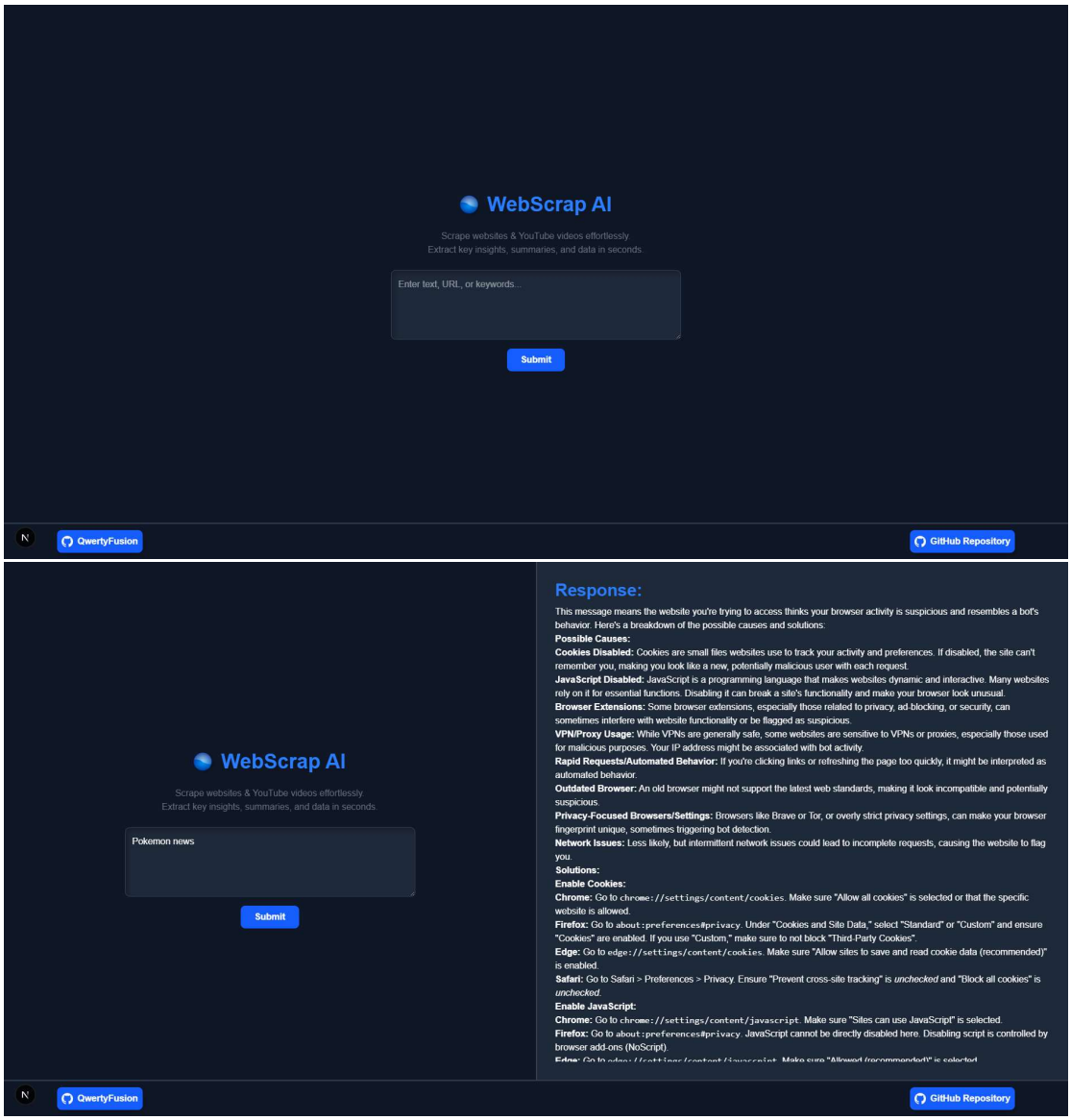
```python
from flask import Flask, request, jsonify
import re
import requests
from bs4 import BeautifulSoup
from youtube_transcript_api import YouTubeTranscriptApi
from duckduckgo_search import DDGS  # Corrected import
import google.generativeai as genai  # Using Gemini API for LLM-based text processing
from dotenv import load_dotenv
import os
from flask_cors import CORS

# Initialize Flask app
app = Flask(__name__)
CORS(app)  # Allow all origins
load_dotenv()  # Load environment variables from .env

# Configure Gemini API
genai.configure(api_key=os.getenv("GEMINI_API_KEY"))

# Function to classify user input
def classify_input(user_text):
    url_pattern = re.compile(r'https?://\S+')
    youtube_pattern = re.compile(r'(https?://)?(www\.)?(youtube\.com|youtu\.?be)/.+')

    urls = url_pattern.findall(user_text)
    youtube_links = [url for url in urls if youtube_pattern.match(url)]

    if youtube_links:
```

page.tsx

frontend > src > app > page.tsx > Home

```tsx
"use client";
import { useState } from "react";
import axios from "axios";
import ReactMarkdown from "react-markdown";
import { motion } from "framer-motion";

Qodo Gen: Options | Test this function
export default function Home() {
  const [inputText, setInputText] = useState("");
  const [responseText, setResponseText] = useState("");
  const [loading, setLoading] = useState(false);
  const [showResponse, setShowResponse] = useState(false);

  const handleSubmit = async () => {
    if (!inputText) return;

    setLoading(true);
    setShowResponse(false); // Hide response initially
    try {
      const response = await axios.post(
        `${process.env.NEXT_PUBLIC_BACKEND_URL}/process`,
        { text: inputText }
      );
      setResponseText(response.data.summary);
      setShowResponse(true); // Show response after fetching
    } catch (error) {
      setResponseText("Error fetching data.");
      setShowResponse(true);
    }
    setLoading(false);
  };

  return (
    <div className="flex flex-col items-start w-full justify-center min-h-screen bg-gray-900 text-white relative">
      {/* Input Section */}
      <motion.div
        className={`flex flex-col items-center justify-center p-6 transition-transform duration-300 ease-in-out ${showResponse ? "w-[50%]" : "w-full"}`}
      >
        <motion.h1
          className="text-4xl font-bold mb-6 text-center text-blue-500 drop-shadow-custom no-select flex items-center justify-center"
          initial={{ opacity: 0, y: -20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ duration: 0.5 }}
        >
          <img src="./icon.png" className="h-8 w-8 mr-4" alt="Developer Profile Link"/>WebScrap AI
        </motion.h1>
        <motion.p
          className="text-md mb-6 text-center text-gray-500 drop-shadow-custom no-select"
          initial={{ opacity: 0, y: -20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ duration: 0.5 }}
        >
          Scrape websites & YouTube videos effortlessly. <br/> Extract key insights, summaries, and data in seconds.
        </motion.p>

        <motion.textarea
          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 focus-within:ring-2 focus-within:ring-blue-500 focus-within:border-blue-500 transition-all duration-300 no-select"
          placeholder="Enter text, URL, or keywords..."
          value={inputText}
          onChange={(e) => setInputText(e.target.value)}
          rows={4}
          initial={{ opacity: 0, scale: 0.9 }}
          animate={{ opacity: 1, scale: 1 }}
          transition={{ duration: 0.4 }}
```

4. Results/Output:-



5. Remarks:-

Signature of the Student

_____

(Name of the Student)

Signature of the Lab Coordinator

_____

(Name of the Coordinator)