LABORATORY REPORT

Application Development Lab (CS33002)

B.Tech Program in ECSc

Submitted By

Name: Shreyaa Venkateswaran

Roll No: 2230120

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

Spring 2024-2025

Table of Content

Experiment Number	5
Experiment Title	Web Scraper using LLMs
Date of Experiment	16-02-2025
Date of Submission	17-03-2025

1. Objective:

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

2. Procedure:

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:

Flask:

```
from flask import Flask, render_template, request

import requests

from bs4 import BeautifulSoup

from groq import Groq
```

```
app = Flask(__name__)

# Initialize Groq client

client =
Groq(api_key="gsk_kK7MeH29PIO69P3LxJ7WWGdyb3FYthtizF
NR062mQ5oHL4gTjYlx")

def scrape_news_article(url):

"""Scrape the text content of a news article."""

response = requests.get(url)

soup = BeautifulSoup(response.text, 'html.parser')

# Extract text from paragraphs (customize based on the website
structure)

paragraphs = soup.find_all('p')

article_text = ' '.join([p.get_text() for p in paragraphs])

return article_text

def summarize_text(text):

"""Summarize the text using Groq API."""

response = client.chat.completions.create(

model="llama-3.3-70b-versatile", # Use a suitable model

messages=[

{"role": "system", "content": "You are a helpful assistant that
summarizes news articles."},

{"role": "user", "content": f"Summarize the following news article
in 3 sentences:\n{text}"}

]

)

return response.choices[0].message.content

@app.route("/", methods=["GET", "POST"])
```

```
def index():

if request.method == "POST":

url = request.form["url"]

try:

# Scrape the article

article_text = scrape_news_article(url)

# Summarize the article

summary = summarize_text(article_text)

return render_template("index.html", summary=summary, url=url)

except Exception as e:

return render_template("index.html", error=str(e))

return render_template("index.html")

if __name__ == "__main__":

app.run(debug=True)
```

index.html:

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width,
initial-scale=1.0">

<title>News Summarizer</title>

<link rel="stylesheet" href="{{ url_for('static', filename='style.css')
}}">

</head>

<body>
```

```
<div class="container">

<h1>News Summarizer</h1>

<form method="POST">

<label for="url">Enter News Article URL:</label>

<input type="text" id="url" name="url"
placeholder="https://example.com/news" required>

<button type="submit">Summarize</button>

</form>

{% if summary %}

<div class="summary">

<h2>Summary</h2>

<p>{{ summary }}</p>

<p><strong>Source:</strong> <a href="{{ url }}"
target="_blank">{{ url }}</a></p>

</div>

{% endif %}

{% if error %}

<div class="error">

<p>Error: {{ error }}</p>

</div>

{% endif %}

</div>

</body>

</html>
```

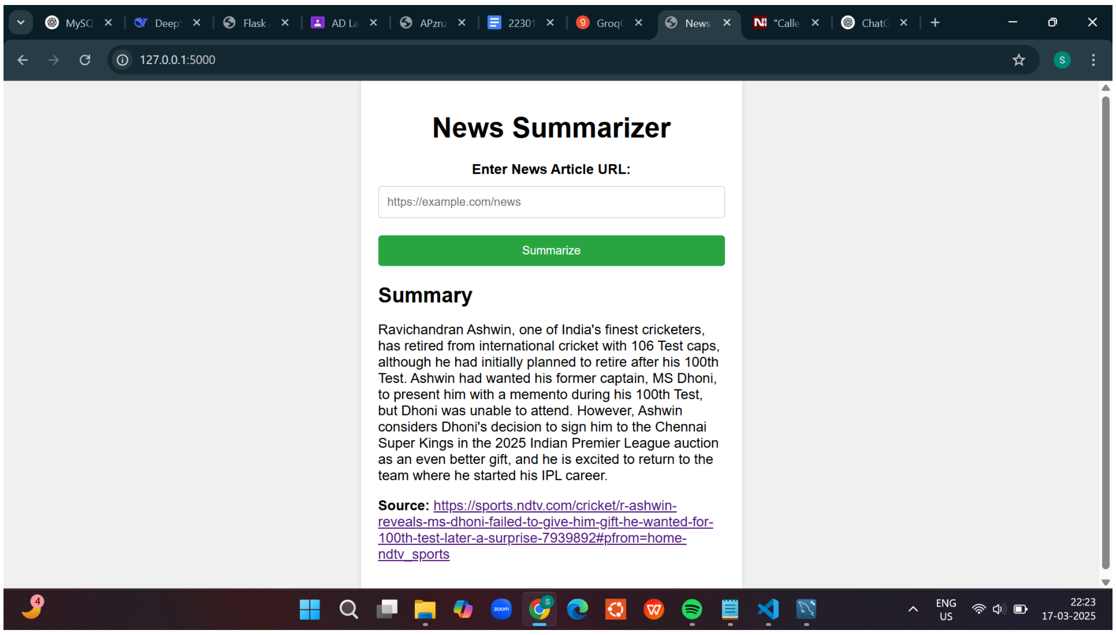
styles.css:

```
        body {
font-family: Arial, sans-serif;
background-color: #f4f4f4;
margin: 0;
padding: 0;
display: flex;
justify-content: center;
align-items: center;
height: 100vh;
}
.container {
background: white;
padding: 20px;
border-radius: 8px;
box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
width: 400px;
text-align: center;
}
h1 {
margin-bottom: 20px;
}
form {
display: flex;
flex-direction: column;
```

```
}
label {
margin-bottom: 10px;
font-weight: bold;
}
input {
padding: 10px;
margin-bottom: 20px;
border: 1px solid #ccc;
border-radius: 4px;
}
button {
padding: 10px;
background-color: #28a745;
color: white;
border: none;
border-radius: 4px;
cursor: pointer;
}
button:hover {
background-color: #218838;
}
.summary, .error {
margin-top: 20px;
text-align: left;
```

```
}
.summary h2 {
margin-bottom: 10px;
}
.error {
color: red;
}
```

4. Results/Output:



News Summarizer

Enter News Article URL:

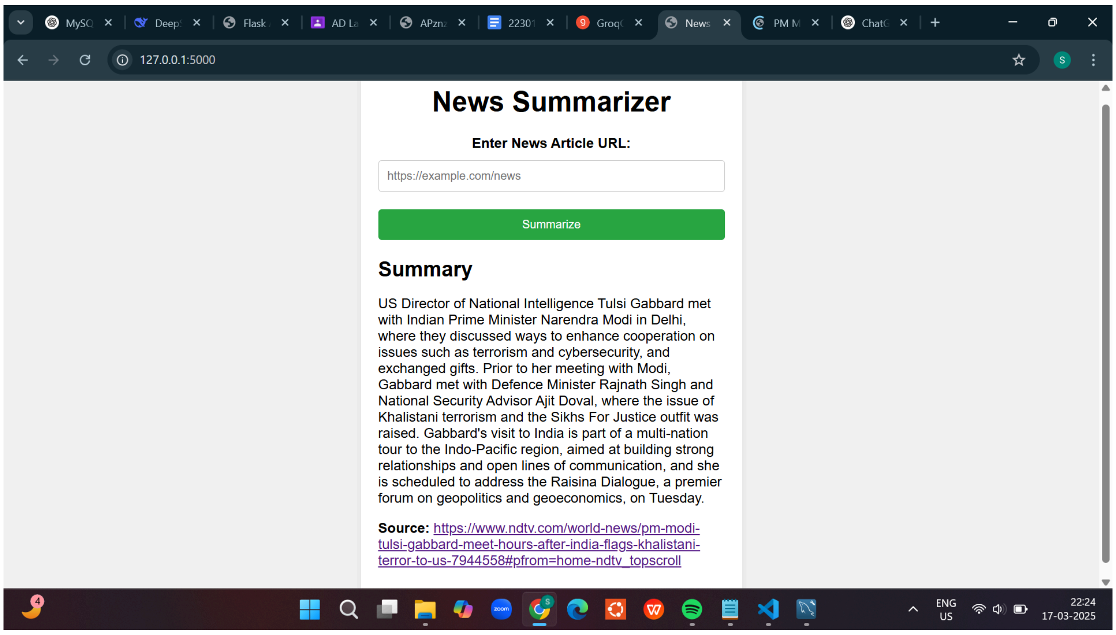
https://example.com/news

Summarize

Summary

Ravichandran Ashwin, one of India's finest cricketers, has retired from international cricket with 106 Test caps, although he had initially planned to retire after his 100th Test. Ashwin had wanted his former captain, MS Dhoni, to present him with a memento during his 100th Test, but Dhoni was unable to attend. However, Ashwin considers Dhoni's decision to sign him to the Chennai Super Kings in the 2025 Indian Premier League auction as an even better gift, and he is excited to return to the team where he started his IPL career.

Source: https://sports.ndtv.com/cricket/r-ashwin-reveals-ms-dhoni-failed-to-give-him-gift-he-wanted-for-100th-test-later-a-surprise-7939892#pfrom=home-ndtv_sports



News Summarizer

Enter News Article URL:

https://example.com/news

Summarize

Summary

US Director of National Intelligence Tulsi Gabbard met with Indian Prime Minister Narendra Modi in Delhi, where they discussed ways to enhance cooperation on issues such as terrorism and cybersecurity, and exchanged gifts. Prior to her meeting with Modi, Gabbard met with Defence Minister Rajnath Singh and National Security Advisor Ajit Doval, where the issue of Khalistani terrorism and the Sikhs For Justice outfit was raised. Gabbard's visit to India is part of a multi-nation tour to the Indo-Pacific region, aimed at building strong relationships and open lines of communication, and she is scheduled to address the Raisina Dialogue, a premier forum on geopolitics and geoeconomics, on Tuesday.

Source: https://www.ndtv.com/world-news/pm-modi-tulsi-gabbard-meet-hours-after-india-flags-khalistani-terror-to-us-7944558#pfrom=home-ndtv_topscroll

5. Remarks:

Built a WebScraper using LLM, model: llama-3.3-70b-versatile using GROQ API. The web page allows a user to enter the url for the website to scrap the information from. After this, the LLM model summarizes the information available in the website and provides it to the user.

Website link: LLM_WebScraper

GitHub link: GitHub

Shreyaa Venkateswaran   Signature of the Lab Coordinator

_______________________   _______________________