### Main Code (app.py)

**#Import libraries**

import streamlit as st

from helper import get\_summary, spacy\_rander, fetch\_news, fetch\_news\_links

**# Set up the configuration for the Streamlit web app**

st.set\_page\_config(

page\_title=”Text Summarization Web App by Kishore,Yaseen,Kisaan ”,

page\_icon=”🔥”,

layout=”wide”,

initial\_sidebar\_state=”expanded”,

menu\_items={

‘About’: ‘This is a header. This is an extremely cool app!’

}

)

**# Display the sidebar title and options**

st.sidebar.title("Text Summarization Web App")

option = ["News Summary and Headlines", "Custom Text Summarization"]

choice = st.sidebar.selectbox("Select your choice", options=option)

**# Custom Text Summarization option**

if choice == "Custom Text Summarization":

st.sidebar.markdown("Copy and paste your text in the text area below to get a summary.")

st.title("Welcome to Custom Text Summarization")

col1, col2 = st.columns(2)

with col1:

text = st.text\_area(label="Enter your text", height=350, placeholder="Enter your text or article here")

if st.button("Get Summary"):

summary = get\_summary(text)

try:

with col2:

st.write("Text Summary:")

st.code(summary)

st.write("Text Headline:")

st.code("Feature Coming Soon")

spacy\_rander(summary)

# Get the original article analysis

# with st.expander("Get Original Article Analysis"):

spacy\_rander(text, text="Yes")

except NameError:

pass

**# News Summary and Headlines option**

if choice == "News Summary and Headlines":

st.title("BBC News Summary")

search\_query = st.text\_input("", placeholder="Enter the topic you want to search")

st.write(" ")

link, title, thumbnail = fetch\_news\_links(search\_query)

fetch\_news = fetch\_news(link)

if link != []:

col1, col2 = st.columns(2)

with col1:

for i in range(len(link)):

if (i % 2) == 0:

st.write(title[i])

with st.expander("Read The Summary"):

st.write(get\_summary(fetch\_news[i]))

st.markdown("[\*\*Read Full Article\*\*]({})".format(link[i]), unsafe\_allow\_

html=True)

st.write(" ")

with col2:

for i in range(len(link)):

if (i % 2) != 0:

st.write(title[i])

with st.expander("Read The Summary"):

st.write(get\_summary(fetch\_news[i]))

st.markdown("[\*\*Read Full Article\*\*]({})".format(link[i]), unsafe\_allow\_

html=True)

st.write(" ")

else:

st.info("No results found for {}. Please try some popular keywords.".format(search

\_query)

**Support code (helper.py)**

**#Import libraries**

import requests

from bs4 import BeautifulSoup

import spacy

from heapq import nshortest

import random

import streamlit as st

**# Function to calculate word frequencies in a document**

def word\_frequency(doc):

word\_frequencies = {}

for word in doc:

**# Check if the word is not a stopword**

if word.text.lower() not in stopwords:

**# Check if the word is not a punctuation**

if word.text.lower() not in punctuation:

**# If the word is not already in the dictionary, add it with a frequency of 1**

if word.text not in word\_frequencies.keys():

word\_frequencies[word.text] = 1

**# If the word is already in the dictionary, increment its frequency by 1**

else:

word\_frequencies[word.text] += 1

return word\_frequencies

**# Function to calculate sentence scores based on word frequencies**

def sentence\_score(sentence\_tokens, word\_frequencies):

sentence\_score = {}

for sent in sentence\_tokens:

for word in sent:

**# Check if the word is in the word frequencies dictionary**

if word.text.lower() in word\_frequencies.keys():

**# If the sentence is not already in the dictionary, add it with the word's frequency as the score**

if sent not in sentence\_score.keys():

sentence\_score[sent] = word\_frequencies[word.text.lower()]

**# If the sentence is already in the dictionary, increment its score by the word's frequency**

else:

sentence\_score[sent] += word\_frequencies[word.text.lower()]

return sentence\_score

**# Function to fetch news links based on a query**

if query == "":

reqUrl = "https://newsapi.org/v2/everything?sources=bbc-news&q=india&language=en&apiKey=ac5568e7ad914659b1d66c0ee6929560".format(news\_api\_key)

else:

reqUrl = "https://newsapi.org/v2/everything?sources=bbc-news&q={}&language=en&apiKey=ac5568e7ad914659b1d66c0ee6929560".format(query, news\_api\_key)

headersList = {

"Accept": "\*/\*",

"User-Agent": "Thunder Client (https://www.thunderclient.com)"

}

payload = ""

response = requests.request("GET", reqUrl, data=payload, headers=headersList).text

response = json.loads(response)

tw = 0

for i in range(len(response["articles"])):

if tw == 10:

pass

else:

if "/news/" in response["articles"][i]["url"] and "stories" not in response["articles"][i]["url"]:

link\_list.append(response["articles"][i]["url"])

title\_list.append(response["articles"][i]["title"])

else:

pass

tw += 1

return link\_list, title\_list, thumbnail\_list

**# Function to fetch news content based on a list of links**

@st.cache(allow\_output\_mutation=False)

def fetch\_news(link\_list):

for i in range(len(link\_list)):

news\_reqUrl = link\_list[i]

headersList = {

"Accept": "\*/\*",

"User-Agent": "Thunder Client (https://www.thunderclient.com)"

}

payload = ""

news\_response = requests.request("GET", news\_reqUrl, data=payload, headers=headersList)

soup = BeautifulSoup(news\_response.content, features="html.parser")

for para in soup.findAll("div", {"data-component":"text-block"}):

news.append(para.find("p").getText())

joinnews = " ".join(news)

news\_list.append(joinnews)

news.clear()

return news\_list

**# Function to generate a summary of a given text**

def get\_summary(text):

doc = nlp(text)

word\_frequencies = word\_frequency(doc)

**# Normalize the word frequencies by dividing each frequency by the maximum frequency**

for word in word\_frequencies.keys():

word\_frequencies[word] = word\_frequencies[word] / max(word\_frequencies.values())

**# Select a percentage of the sentences with the highest scores to form the summary**

select\_length = int(len(sentence\_tokens) \* 0.10)

summary = nlargest(select\_length, sentence\_scores, key=sentence\_scores.get)

summary = [word.text for word in summary]

summary = " ".join(summary)

return summary