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".for
mat(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
```