**Worko AI-ML Assignment**

Sakshi Yagnik

**Scenario – III**

Build a simple research assistant that can answer questions using web search + LLM synthesis.

**Requirements:**

* Take a user question
* Break it down into search queries
* Use a search API to gather information
* Use an LLM to synthesize findings into a coherent answer
* Cite sources appropriately
* Handle cases where information is insufficient

**Solution:**

**High-Level Workflow –**

(User Question ➝ Query Breakdown ➝ Web Search ➝ Info Extraction ➝ LLM Synthesis ➝ Final Answer with Citations)

**Tech Stack (for implementation)**

* **Language**: Python
* **Search API**: SerpAPI / Bing Web Search API / Google Programmable Search
* **LLM**: OpenAI GPT-4 (via API)

## Optional Libraries: requests, openai, bs4, tiktoken, langchain (optional for orchestration) Step-by-Step Logic

**1. Take a User Question**

user\_question = input("Enter your research question: ")

**2. Break Into Search Queries:** Use keyword extraction or heuristics:

**from** nltk.tokenize **import** word\_tokenize

**from** nltk.corpus **import** stopwords

**def** **extract\_search\_queries**(question):

tokens = word\_tokenize(question)

keywords = [word **for** word **in** tokens if word.lower() **not in** stopwords.words('english') **and** word.isalnum()]

**return** [" ".join(keywords)]

**3. Use a Search API**

**import** requests

**def search\_web**(query):

api\_key = "YOUR\_SERPAPI\_KEY"

params = {

"q": query,

"api\_key": api\_key,

"engine": "google"

}

response = requests.get("https://serpapi.com/search", params=params)

results = response.json()

**return** results['organic\_results'][:5] # top 5 results

**4. Extract Snippets and Links**

**def extract\_content**(search\_results):

info = ""

citations = []

**for** result **in** search\_results:

title = result.get('title')

snippet = result.get('snippet')

link = result.get('link')

**if** snippet:

info += f"{title}: {snippet}\n"

citations.append(link)

**return** info, citations

**5. LLM Synthesis**

**import** openai

**def synthesize\_answer**(user\_question, context):

openai.api\_key = "YOUR\_OPENAI\_API\_KEY"

prompt = f"""You are a helpful research assistant. Use the following information to answer the user's question:

Context:

{context}

Question: {user\_question}

Answer (with citations where appropriate):"""

response = openai.ChatCompletion.create(

model="gpt-4",

messages=[{"role": "user", "content": prompt}],

temperature=0.7

)

**return** response['choices'][0]['message']['content']

**6. Full Pipeline**

**def research\_assistant**(question):

queries = extract\_search\_queries(question)

all\_results = []

**for** query **in** queries:

search\_results = search\_web(query)

all\_results.extend(search\_results)

if **not** all\_results:

**return** "Sorry, not enough information was found to answer your question."

content, sources = extract\_content(all\_results)

answer = synthesize\_answer(question, content)

**return** answer + "\n\nSources:\n" + "\n".join(sources)

**Sample Usage**

question = "What are the health benefits of intermittent fasting?"

**print**(research\_assistant(question))

**Output Example**

**Answer**:  
Intermittent fasting has been shown to improve metabolic health, aid in weight loss, and may enhance brain function. It can help regulate insulin levels, reduce inflammation, and support cellular repair processes. However, effects vary by individual.

**Sources**:

* <https://www.healthline.com/nutrition/intermittent-fasting-guide>
* <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5783752/>

**Fallback: Insufficient Information**

Add a simple fallback inside research\_assistant():

**if not** content:

**return** "I couldn't find enough reliable information to answer your question. Try rephrasing it or asking something else."