Name	UB Number
Vaishnavi Varanasi	50590668

Project Update - MS Practicum MGS 649 SEM

Project Title: UB Research Assistant

Introduction:

The **UB Research Assistant** is an AI-powered academic tool designed specifically for the University at Buffalo community. It simplifies the research process for students, faculty, and scholars by offering quick paper summarization, smart recommendations, and a collaboration matching system with UB faculty. By combining the power of language models, semantic search, and open-access APIs, the tool helps users discover, digest, and connect around cutting-edge academic work.

Objectives:

- Automate the **summarization of research papers** using LLMs.
- Provide **real-time recommendations** of similar papers.
- Enable users to **find UB faculty members** working on related research areas.
- Reduce time spent on reading and searching through disconnected platforms.
- Lay the groundwork for future global collaboration beyond UB.

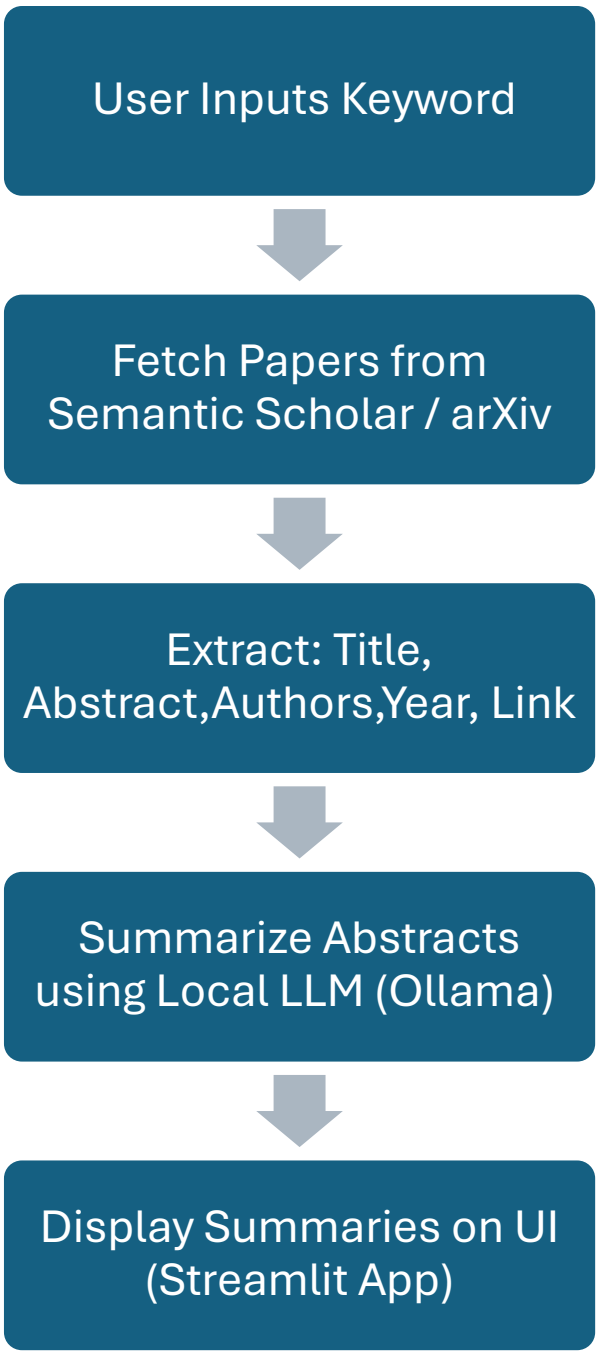
Scope:

1. **Research Paper Search**
   – Via Semantic Scholar and arXiv APIs.
2. **Paper Summarization**
   – Using a locally served language model (Mistral via Ollama).
3. **Paper Recommendation Engine**
   – Based on keyword relevance and embeddings.

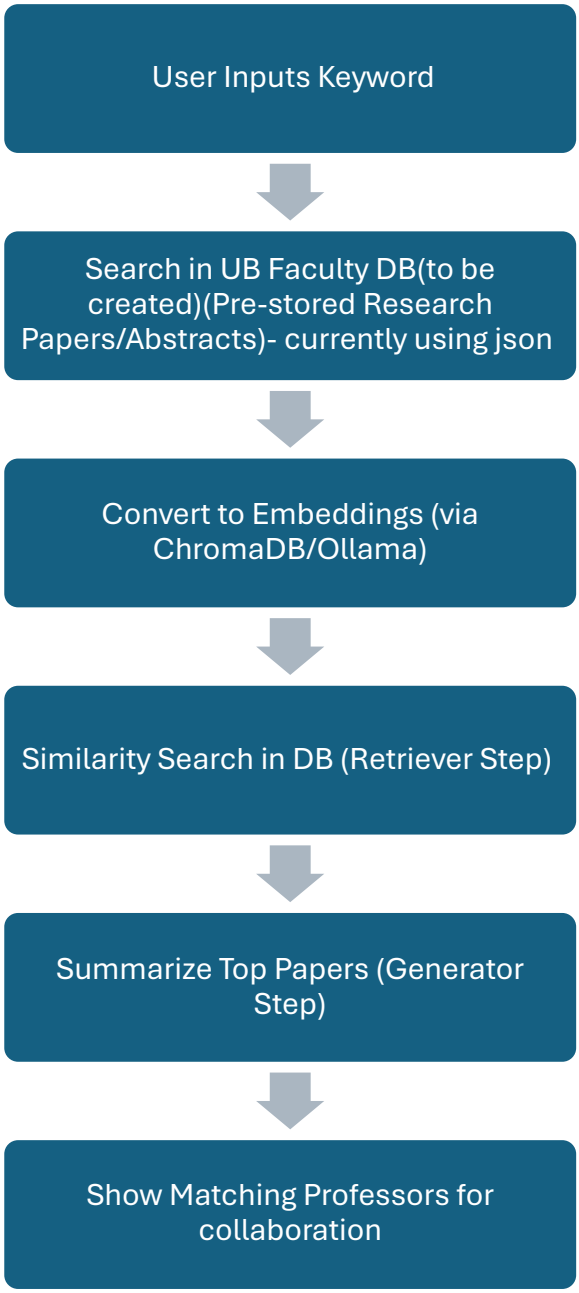
4. **UB Faculty Research Matcher**

   – Matches user keywords with UB faculty research stored in a local JSON DB and ChromaDB.

Part A- LIVE SEARCH & ABSTRACT SUMMARIZATION



Part B- UB PROFESSORS' DATABASE & RAG COLLABORATION MATCHER



End-User Guide:

Step 1	Visit Streamlit App URL
Step 2	Enter Research keyword/topic
Step 3	In Part A- Fetch Papers Live -> Summarize and Display
Step 4	In Part B -> Search UB Faculty DataBase, Retrieve related Papers, Summarize, Show matching Professors

Technical Details of the Project:

1. Python- Version 3.11- Needed for compatibility with modern libraries like ChromaDB, Streamlit, and Ollama
2. Libraries Installed and their purpose-

Library	Purpose
Streamlit	Frontend UI for easy, fast app deployment.
Chromadb	Vector database for embedding based retrieval
Requests	Fetch papers from extrenal APIs
Ollama	Local LLM runtime without internet

3. APIS used-

API	Usage
Semantic Scholar	Highly relevant academic papers, structured metadata, free access for research
arXiv	Fetches open-access research papers across science, tech, and mathematics.

4. UI Framework choice(Streamlit)- It is extremely fast to develop and deploy a full-fledged web app with minimal Python code. It also auto-generates UI components (text boxes, buttons, tables) without needing HTML/CSS/JS knowledge. It is perfect for MVPs and internal academic tools.

Code:

Research_Summary_App_Final.py - /Users/vaishu/Documents/ub_env/Researc...

```
import streamlit as st
import requests
import xml.etree.ElementTree as ET
import json
from datetime import datetime
import pandas as pd
import os
```

These imports load necessary Python libraries:

- streamlit: Builds the web app UI.
- requests: Makes API calls to fetch paper data.
- xml.etree.ElementTree: Parses XML from arXiv.
- json: Loads/saves JSON files.
- datetime: Timestamps feedback entries.
- pandas: For storing/handling feedback data.
- os: File path checking.

```
# ---------------------- CONFIG ----------------------
OLLAMA_MODEL = "mistral"
OLLAMA_URL = "http://localhost:11434/api/generate"
S2_URL = "https://api.semanticscholar.org/graph/v1/paper/search"
ARXIV_URL = "http://export.arxiv.org/api/query"
```

Defines constants:

- The model name for Ollama (LLM).
- Local API endpoint for generating summaries.
- Semantic Scholar and arXiv endpoints for paper data.

```
# ---------------------- PAGE CONFIG ----------------------
st.set_page_config(page_title="UB Research Assistant", layout="wide")
```

Sets the browser tab title and ensures full-width layout.

```
# ---------------------- HEADER ----------------------
st.markdown("""
    <div class="ub-header">
        <h1>UB Research Assistant</h1>
        <p>The UB way to research smarter</p>
    </div>
""", unsafe_allow_html=True)
```

Displays the blue header with project title and tagline.

```
# ---------------------- SIDEBAR ----------------------
with st.sidebar:
    st.markdown("""
        <div class="sidebar-content">
            <div class="sidebar-header">About This Assistant</div>
            <p>
                This tool acts as a personalized research assistant to students,
                <ul>
                    <li>Summarize recent papers</li>
                    <li>Explore UB faculty research</li>
                    <li>Find collaboration opportunities</li>
                </ul>
            </p>
            <div class="sidebar-header">Quick Links</div>
            <p>
                <a href="https://www.buffalo.edu/" target="_blank">UB Homepage</
                <a href="https://library.buffalo.edu/" target="_blank">UB Librar
                <a href="https://csed.buffalo.edu/" target="_blank">Dept. of Com
            </p>
            <div class="sidebar-header">Feedback</div>
            <p>We'd love to hear from you!</p>
        </div>
    """, unsafe_allow_html=True)

    with st.form("feedback_form"):
        name = st.text_input("Please enter your name")
        email = st.text_input("Email")
        feedback_text = st.text_area("Your Feedback", height=100)
        submitted = st.form_submit_button("Submit")

        if submitted and name.strip() and feedback_text.strip() and email.strip(
            timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            new_entry = pd.DataFrame([[timestamp,name.strip(), email.strip(), fe
                                       columns=["Timestamp","Name","Email", "Feedb
            file_path = "feedback_log.csv"
            if os.path.exists(file_path):
                existing = pd.read_csv(file_path)
                updated = pd.concat([existing, new_entry], ignore_index=True)
            else:
```

Ln: 1 Col: 0

Sidebar shows:

- What the assistant does
- Quick links to UB resources
- A feedback form (name, email, message)

```
# ---------------------- LOAD UB PAPERS ----------------------
@st.cache_data
def load_ub_papers():
    with open("ub_papers.json", "r") as f:
        return json.load(f)

# ---------------------- OLLAMA SUMMARIZATION ----------------------
```

- Loads UB research papers from a local JSON file.
- Uses Streamlit caching for efficiency.

```
# ---------------------- OLLAMA SUMMARIZATION ----------------------
def summarize_text(text):
    prompt = f"Summarize the following research abstract in 2 simple sentences:\
    response = requests.post(OLLAMA_URL, json={
        "model": OLLAMA_MODEL,
        "prompt": prompt,
        "stream": False
    })
    return response.json().get("response", "(No summary returned)").strip()

# ---------------------- OLLAMA RELEVANCE EXPLANATION ----------------------
```

- Sends abstract text to Ollama LLM API.
- Returns a 2-sentence summary.

```
# ---------------------- OLLAMA RELEVANCE EXPLANATION ----------------------
def get_relevance_reason(paper, keyword):
    prompt = (
        f"Given the following research paper and a keyword, explain in 1 line wh
        f"to that keyword.\n\n"
        f"Keyword: {keyword}\n\n"
        f"Title: {paper['title']}\n"
        f"Abstract: {paper['abstract']}\n\n"
        f"Respond with a single sentence only."
    )
    try:
        response = requests.post(OLLAMA_URL, json={
            "model": OLLAMA_MODEL,
            "prompt": prompt,
            "stream": False
        })
        return response.json().get("response", "(No response)").strip()
    except Exception as e:
        return f"(Ollama error: {str(e)})"

# ---------------------- SEMANTIC SCHOLAR FETCH ----------------------
```

- Sends paper details and user keyword to LLM.
- Returns 1-line explanation of why the paper is relevant.

```python
# ---------------------- SEMANTIC SCHOLAR FETCH ----------------------
def fetch_semantic_scholar(query):
    res = requests.get(S2_URL, params={
        "query": query,
        "fields": "title,abstract,url,authors,year",
        "limit": 5
    })
    return [
        {
            "title": p["title"],
            "abstract": p.get("abstract", ""),
            "url": p.get("url", ""),
            "authors": ", ".join([author["name"] for author in p.get("authors",
            "year": p.get("year", "N/A")
        } for p in res.json().get("data", []) if p.get("abstract")
    ]

# ---------------------- ARXIV FETCH ----------------------
```

- Sends query to Semantic Scholar.
- Returns top 5 papers with title, abstract, authors, year, and link.

```python
# ---------------------- ARXIV FETCH ----------------------
def fetch_arxiv(query):
    url = f"{ARXIV_URL}?search_query=all:{query}&start=0&max_results=3"
    res = requests.get(url)
    root = ET.fromstring(res.content)
    ns = {'atom': 'http://www.w3.org/2005/Atom'}
    papers = []
    for entry in root.findall('atom:entry', ns):
        authors = [author.find('atom:name', ns).text for author in entry.findall
        published = entry.find('atom:published', ns).text if entry.find('atom:pu
        year = published[:4] if published != "N/A" else "N/A"
        papers.append({
            "title": entry.find('atom:title', ns).text.strip(),
            "abstract": entry.find('atom:summary', ns).text.strip(),
            "url": entry.find('atom:id', ns).text.strip(),
            "authors": ", ".join(authors),
            "year": year
        })
    return papers
```

- Sends query to arXiv.
- Parses XML to extract papers with title, abstract, author names, year, and link.

```python
# ---------------------- MAIN TABS ----------------------
summary_tab, ub_tab, collab_tab = st.tabs(["Summary", "UB Database", "Collaborat
```

Creates three tabs:

- Summary: Paper summarizer
- UB Database: Search UB-specific research

- Collaboration Finder: Find professors doing similar work

```
# ---------- TAB 1: Summary ----------
with summary_tab:
    st.markdown("### Research Summarizer")
    topic = st.text_input("Enter a keyword to fetch and summarize papers")

    if topic:
        st.info("Fetching papers from Semantic Scholar and arXiv...")
        papers = fetch_semantic_scholar(topic) + fetch_arxiv(topic)

        for paper in papers:
            summary = summarize_text(paper['abstract'])
            st.markdown(f"""
                <div class="paper-container">
                    <div class="paper-title">{paper['title']}</div>
                    <div class="paper-meta">
                        Authors: {paper['authors']}<br>
                        Year: {paper['year']}<br>
                        <a href="{paper['url']}" target="_blank">View Full Paper
                    </div>
                    <details><summary>Abstract</summary>
                    <p class="paper-abstract">{paper['abstract']}</p>
                    </details>
                    <div class="relevance-box"><b>AI Summary:</b> {summary}</div
                </div>
            """, unsafe_allow_html=True)
```

- Lets user input a keyword
- Fetches papers from both APIs
- Uses LLM to summarize
- Displays output as paper cards with links and AI summaries

```python
# ---------- TAB 2: UB Paper Database ----------
with ub_tab:
    st.markdown("### UB Research Paper Archive")
    st.markdown("Search and view relevant UB faculty research papers.")

    keyword = st.text_input("Enter a research keyword", key="ub_search")

    if keyword:
        ub_papers = load_ub_papers()
        filtered = []

        with st.spinner("Finding relevant UB papers..."):
            for paper in ub_papers:
                if keyword.lower() in paper['title'].lower() or keyword.lower()
                    explanation = get_relevance_reason(paper, keyword)
                    paper["reason"] = explanation
                    filtered.append(paper)

        if filtered:
            st.markdown(f"**{len(filtered)} relevant papers found:**")
            for paper in filtered:
                st.markdown(f"""
                    <div class="paper-container">
                        <div class="paper-title">{paper["title"]}</div>
                        <div class="paper-meta">
                            Authors: {', '.join(paper['authors'])}<br>
                            Department: {paper['department']}<br>
                            Email: {paper['email']}<br>
                            Year: {paper['year']}<br>
```

Ln: 1 Col: 0

- Searches the UB research database for matches to a keyword
- Shows paper details and an explanation of relevance

```python
# ---------- TAB 3: Collaboration Finder ----------
with collab_tab:
    st.markdown("### UB Professor Collaboration Finder")
    st.markdown("Search for UB professors doing similar research.")

    keyword = st.text_input("Enter a research keyword", key="collab_input")

    if keyword:
        st.session_state["last_keyword"] = keyword
        ub_papers = load_ub_papers()
        results = [p for p in ub_papers if keyword.lower() in p["title"].lower()
        if results:
            st.markdown(f"**{len(results)} matches found:**")
            for paper in results:
                explanation = get_relevance_reason(paper, keyword)
                st.markdown(f"""
                    <div class="paper-container">
                        <div class="paper-title">{paper["title"]}</div>
                        <div class="paper-meta">
                            Authors: {', '.join(paper['authors'])}<br>
                            Department: {paper['department']}<br>
                            Email: {paper['email']}<br>
                            <a href="{paper['link']}" target="_blank">View Full
                        </div>
                        <div class="relevance-box"><b>Why Relevant:</b> {explana
                    </div>
                """, unsafe_allow_html=True)
        else:
            st.warning("No matching faculty found.")
    else:
        st.info("Enter a keyword to begin matching.")
```

- Matches user input keyword with UB faculty research
- Suggests professors and displays their work

Made it a Portfolio Project:

1. GitHub Repository
    a. Full source code (Streamlit + Python)
    b. CSS customization
    c. API integration (Semantic Scholar + arXiv)
    d. Sample UB faculty JSON database
2. Deployment Method

    Local Deployment:

    a. Used streamlit run app.py to run locally
    b. Ollama runs locally for LLM summaries

Public URL:

- Deployed using Streamlit Cloud
- Steps:
    1. Pushed code to GitHub
    2. Connected to Streamlit Cloud
    3. Set up secrets (API key)
    4. Shared app via public URL: https://research-assistant-at-ub.streamlit.app

Challenges:

- Selecting the most flexible and open API (Semantic Scholar vs. arXiv)
- Handling abstract-less papers
- Designing a clean yet informative UI
- Limiting public access due to OpenAI's API rate limits

Screenshots

Set1:

Testing Semantic Scholar API:

```
# 📦 Import the requests library to make API calls
import requests
```

This line imports the requests library, which lets Python code talk to websites or APIs.

```
# 🧠 Define a function that will search papers on Semantic Scholar
def fetch_from_semantic_scholar(query):
    print("\n📚 SEMANTIC SCHOLAR RESULTS\n" + "-" * 40)

    # 🔗 Semantic Scholar API endpoint
    url = "https://api.semanticscholar.org/graph/v1/paper/search"

    # ⚙️ Parameters for the search
    params = {
        "query": query,  # 🗣️ Topic or keyword entered by the user
        "limit": 5,  # 📄 Number of papers to fetch
        "fields": "title,abstract,authors,year,url"  # 🧾 What info we want back
    }
```

```
🔍 Enter your research topic: Applications of AI in Marketing Analytics

📚 SEMANTIC SCHOLAR RESULTS
----------------------------------------
1. 📘 Title: Opportunities for the Use of AI in Marketing Communication by Educational Institutions
   🧠 Abstract: Artificial intelligence has become an integral tool in various sectors, including marketing and marketing communication, offering unprecedented opportunities for enh
ancing efficiency and creativity. In marketing, AI's capabilities extend from automating routine tasks to providing deep insights throu...
   👨‍🔬 Authors: ['Ladislav Pátík', 'Jana Galera Matúšová', 'Katarína Načiniaková']
   📅 Year: 2024
   🔗 URL: https://www.semanticscholar.org/paper/d3c8f3ef5dac7b8564bf086b2edb02c24a82dd34
----------------------------------------
2. 📘 Title: Artificial Intelligence in Marketing Communication: A Comprehensive Exploration of the Integration and Impact of AI
   🧠 Abstract: This study investigates the transformative impact of artificial intelligence (AI) on marketing communications through an evaluation research approach. Focused on enh
ancing personalization, efficiency, and strategic insight, the study explores AI's applications in content creation, customer service, ...
   👨‍🔬 Authors: ['Hafize Nurgül Durmuş Şenyapar']
   📅 Year: 2024
   🔗 URL: https://www.semanticscholar.org/paper/0789e46115612436e37526dc7c6f8f644b576f69
----------------------------------------
3. 📘 Title: From analytics to empathy and creativity: Charting the AI revolution in marketing practice and education
   🧠 Abstract: The rapid advancement of artificial intelligence (AI) increasingly demands an understanding of its impact on marketing practice and education. Our hybrid literature
review synthesizes 312 peer-reviewed articles on AI in marketing and consumer behavior, using scientometrics and the TCCM (Theory, Cont...
   👨‍🔬 Authors: ['Giulia Pavone', 'Lars Meyer-Waarden', 'Andreas Munzel']
   📅 Year: 2024
   🔗 URL: https://www.semanticscholar.org/paper/43b901138fe4994ed58d700e7661aa426f32568a
----------------------------------------
4. 📘 Title: The Integration of AI in Digital Marketing: Opportunities and Challenges
   🧠 Abstract: The integration of Artificial Intelligence (AI) in digital marketing has revolutionized the way businesses interact with customers and optimize their marketing strat
egies. AI technologies, such as machine learning, natural language processing, and predictive analytics, have enabled marketers to deli...
   👨‍🔬 Authors: ['Tejaswini Bastray']
   📅 Year: 2025
   🔗 URL: https://www.semanticscholar.org/paper/aa852efdaaf07cf181e47479f0da96c6edf628ba
----------------------------------------
5. 📘 Title: The impact of AI-supported marketing capabilities and analytics on SMEs' customer agility and marketing performance
   🧠 Abstract: This research examines the impact of marketing analytics and artificial intelligence applications on customer agility and marketing performance in businesses that ad
opt e-commerce. In this quantitative study, data were collected through a questionnaire. Data collected from 227 managers online were a...
   👨‍🔬 Authors: ['Fatma Demirağ']
   📅 Year: 2025
   🔗 URL: https://www.semanticscholar.org/paper/37decead7ce84963b9a094b6dc8fea57af0d66b8
----------------------------------------
(rag-env) (base) VaishnavisMBP2:rag-env vaishu$
```

Set2:

```python
# Insert documents into ChromaDB
collection.add(
    documents=documents,
    metadatas=metadatas,
    ids=ids
)

print("✅ Abstracts stored in ChromaDB")
```

Set3:

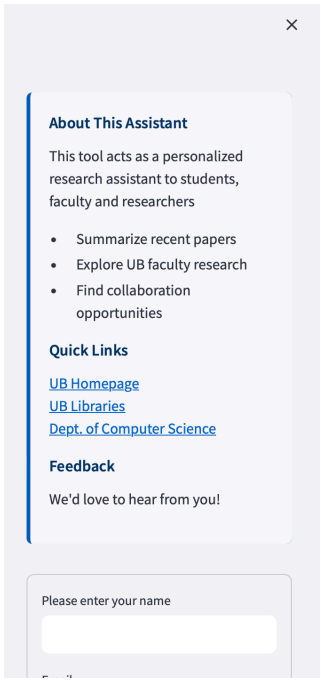
Live Summary Generator In Terminal:

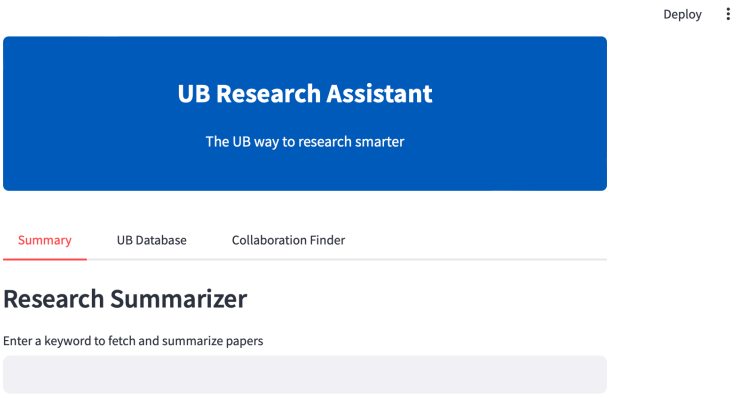
```python
import requests
import xml.etree.ElementTree as ET

OLLAMA_URL = "http://localhost:11434/api/generate"
OLLAMA_MODEL = "mistral"
QUERY = input("🔍 Enter a research topic:\n> ")
MAX_RESULTS = 3

# 🔷 Step 1: Semantic Scholar
def fetch_semantic_scholar(query):
    print("\n📡 Fetching from Semantic Scholar...")
    url = "https://api.semanticscholar.org/graph/v1/paper/search"
    params = {
        "query": query,
        "fields": "title,abstract,url",
        "limit": MAX_RESULTS
    }
    try:
        res = requests.get(url, params=params).json()
        return [{
            "source": "Semantic Scholar",
            "title": p["title"],
            "abstract": p.get("abstract", ""),
            "url": p.get("url", "")
        } for p in res.get("data", []) if p.get("abstract")]
    except:
        return []

# 🔷 Step 2: arXiv
def fetch_arxiv(query):
    print("📡 Fetching from arXiv...")
    url = f"http://export.arxiv.org/api/query?search_query=all:{query}&start=0&max_results={MAX_RESULTS}"
    res = requests.get(url)
    root = ET.fromstring(res.content)
    ns = {'atom': 'http://www.w3.org/2005/Atom'}
    papers = []
    for entry in root.findall('atom:entry', ns):
        title = entry.find('atom:title', ns).text.strip()
        summary = entry.find('atom:summary', ns).text.strip()
        link = entry.find('atom:id', ns).text.strip()
        papers.append({"source": "arXiv", "title": title, "abstract": summary, "url": link})
    return papers

# 🔷 Step 3: CrossRef
def fetch_crossref(query):
    print("📡 Fetching from CrossRef...")
    url = f"https://api.crossref.org/works?query={query}&rows={MAX_RESULTS}"
    try:
        res = requests.get(url).json()
        return [{
            "source": "CrossRef",
            "title": item["title"][0],
            "abstract": "",  # No abstract support
            "url": item.get("URL", "")
        } for item in res["message"]["items"] if "title" in item]
    except:
        return []

# 🧠 Step 4: Summarize with Ollama
def summarize_abstract(title, abstract, url, source):
```

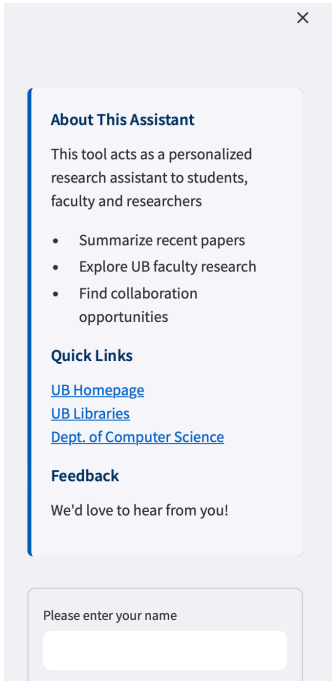
Ln: 9 Col: 22

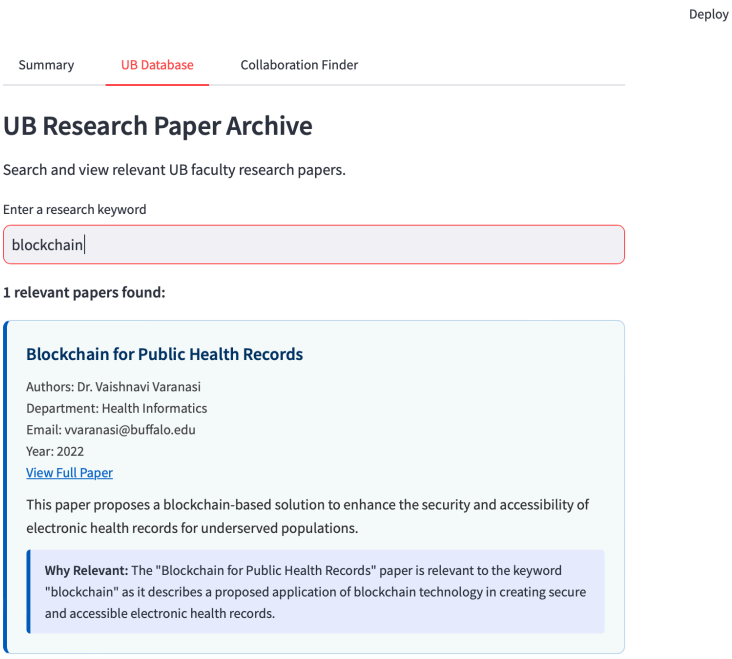
```
📚 Source   : arXiv
📌 Title    : Bringing order into the realm of Transformer-based language models for
  artificial intelligence and law
📝 Summary : 1. Transformer-based Language Models (TLMs) are leading advancements in deep learning solutions, particularly in natural language processing and understanding. They hav
e significantly improved AI approaches in various legal tasks.
2. This article provides a comprehensive overview of TLM-based methods applied to AI problems in the legal field. It aims to understand how Transformers contribute to AI support in
legal processes, while also identifying current limitations and opportunities for further research.
🔗 Link     : http://arxiv.org/abs/2308.05502v2

⏭️Skipping CrossRef paper 'Artifical Intelligence in Trauma' (no abstract).

⏭️Skipping CrossRef paper 'NIHR Research: Artifical Intelligence' (no abstract).

⏭️Skipping CrossRef paper 'Intelligence, Artifical' (no abstract).

(rag-fresh-env) (base) VaishnavisMBP2:rag-fresh-env vaishu$ Computer Netwroks
-bash: Computer: command not found
(rag-fresh-env) (base) VaishnavisMBP2:rag-fresh-env vaishu$ python live_summary_multiple_APIs.py
🔍 Enter a research topic:
> Computer Networks

📡 Fetching from Semantic Scholar...
📡 Fetching from arXiv...
📡 Fetching from CrossRef...

🧠 Summarizing 6 papers using Ollama...

📚 Source   : arXiv
📌 Title    : Computing Power Network: A Survey
📝 Summary : 1. A new networking paradigm called Computing Power Network (CPN) is proposed to leverage distributed computing resources in the era of cloud computing, edge computing,
 and smart devices. This network aims to connect heterogeneous resources for flexible scheduling.

2. This research survey covers the state-of-the-art on CPN, discussing its architecture, advantages, issues such as modeling, resource allocation, network forwarding, and more. It a
lso presents a testbed evaluation, use cases, enabling technologies, and open challenges for future research.
🔗 Link     : http://arxiv.org/abs/2210.06080v2

📚 Source   : arXiv
📌 Title    : Analysis of network by generalized mutual entropies
📝 Summary : 1. A new method called generalized mutual entropy is proposed to analyze complex network structures, which has been tested on computer-simulated scale-free networks, ra
ndom networks, and their combinations.
2. Potential real-world applications of this method in network analysis are explored.
🔗 Link     : http://arxiv.org/abs/0709.0929v1

📚 Source   : arXiv
📌 Title    : A Multilayer Model of Computer Networks
📝 Summary : 1. This research proposes a formal model of computer networks using hierarchical multilayer networks, where each layer is represented as a multiplex network.
2. The study also identifies the essential set of layers needed for network architecture representation when considering the system methodology in network analysis.
🔗 Link     : http://arxiv.org/abs/1509.00721v1

⏭️Skipping CrossRef paper 'Computer networks, the journal, and computer networks, the technology' (no abstract).

⏭️Skipping CrossRef paper 'Computer Networks: Publisher's Note' (no abstract).

⏭️Skipping CrossRef paper 'Local computer networks' (no abstract).

(rag-fresh-env) (base) VaishnavisMBP2:rag-fresh-env vaishu$
```

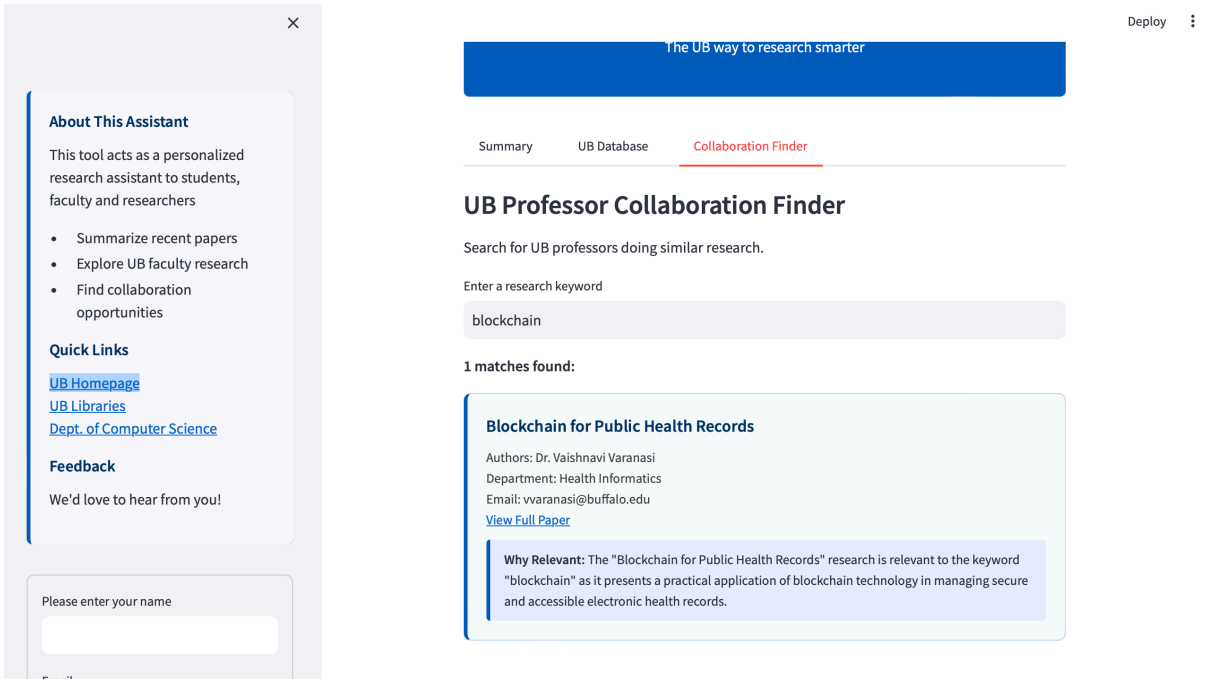
In Streamlit:

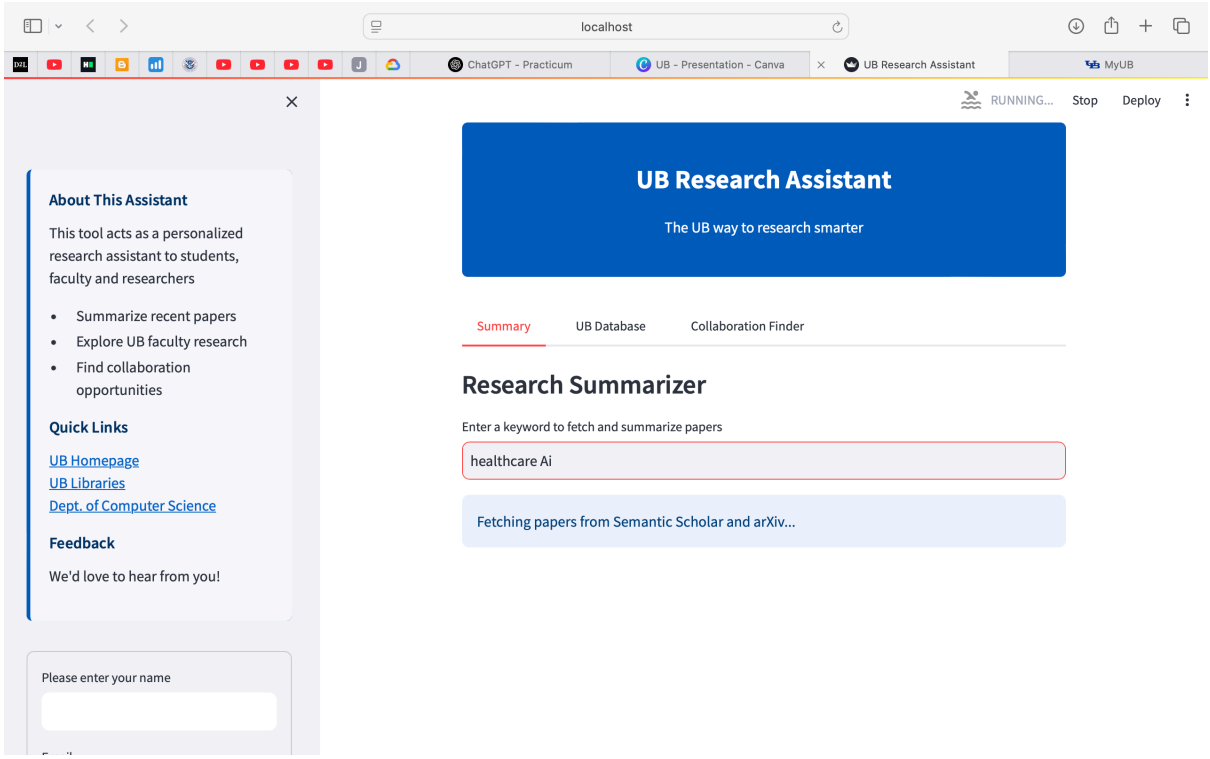


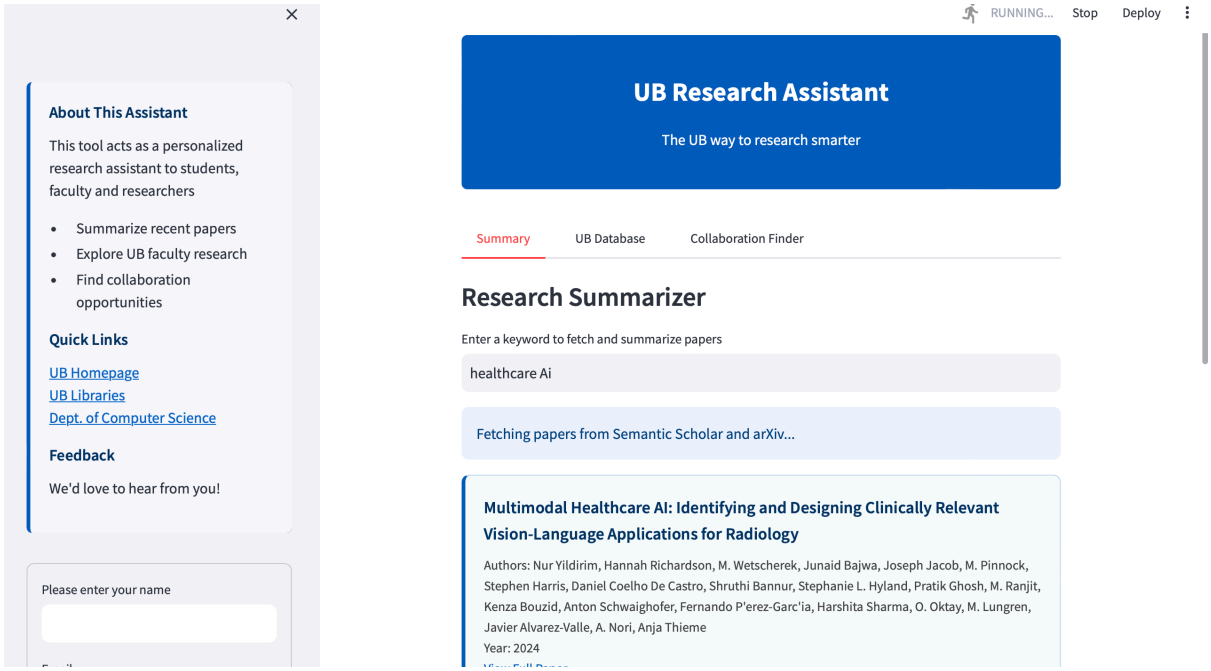


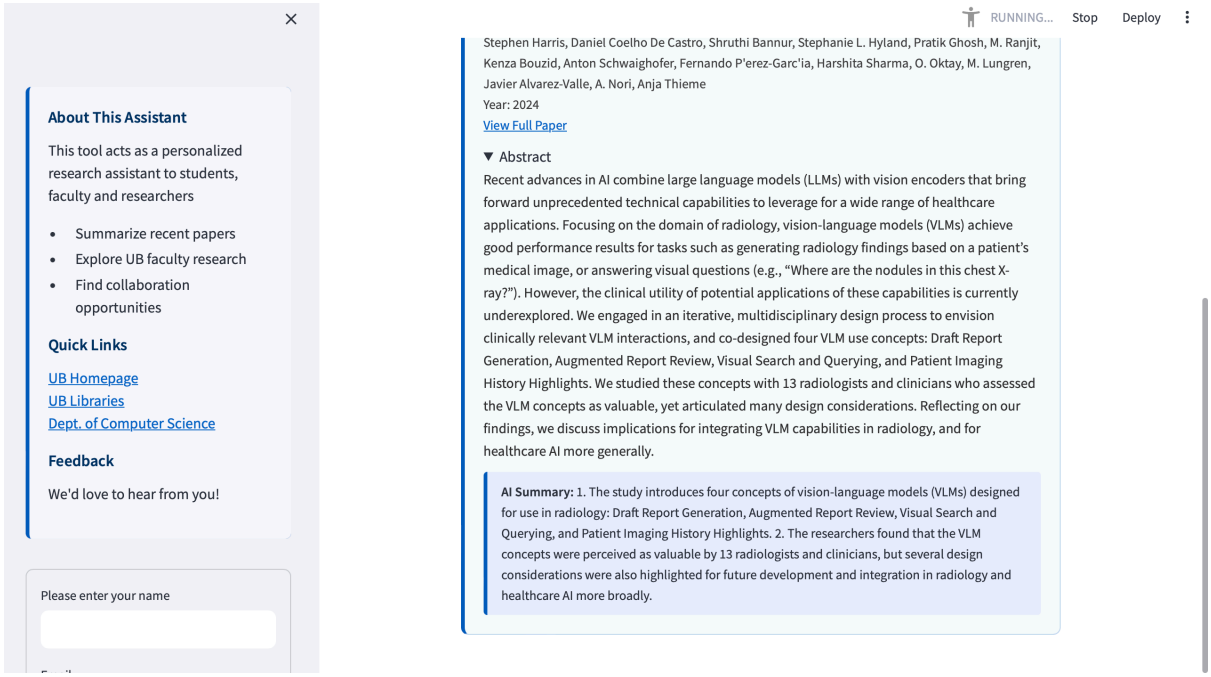





Deploy
The UB way to research smarter
Summary
UB Database
Collaboration Finder
UB Professor Collaboration Finder
Search for UB professors doing similar research.
Enter a research keyword
blockchain
1 matches found:
Blockchain for Public Health Records
Authors: Dr. Vaishnavi Varanasi
Department: Health Informatics
Email: vvaranasi@buffalo.edu
View Full Paper
Why Relevant: The "Blockchain for Public Health Records" research is relevant to the keyword "blockchain" as it presents a practical application of blockchain technology in managing secure and accessible electronic health records.
About This Assistant
This tool acts as a personalized research assistant to students, faculty and researchers
Summarize recent papers
Explore UB faculty research
Find collaboration opportunities
Quick Links
UB Homepage
UB Libraries
Dept. of Computer Science
Feedback
We'd love to hear from you!
Please enter your name


localhost
ChatGPT - Practicum
UB - Presentation - Canva
UB Research Assistant
MyUB
RUNNING...
Stop
Deploy
UB Research Assistant
The UB way to research smarter
Summary
UB Database
Collaboration Finder
Research Summarizer
Enter a keyword to fetch and summarize papers
healthcare Ai
Fetching papers from Semantic Scholar and arXiv...
About This Assistant
This tool acts as a personalized research assistant to students, faculty and researchers
Summarize recent papers
Explore UB faculty research
Find collaboration opportunities
Quick Links
UB Homepage
UB Libraries
Dept. of Computer Science
Feedback
We'd love to hear from you!
Please enter your name


RUNNING...
Stop
Deploy
About This Assistant
This tool acts as a personalized research assistant to students, faculty and researchers
Summarize recent papers
Explore UB faculty research
Find collaboration opportunities
Quick Links
UB Homepage
UB Libraries
Dept. of Computer Science
Feedback
We'd love to hear from you!
Please enter your name
UB Research Assistant
The UB way to research smarter
Summary
UB Database
Collaboration Finder
Research Summarizer
Enter a keyword to fetch and summarize papers
healthcare Ai
Fetching papers from Semantic Scholar and arXiv...
Multimodal Healthcare AI: Identifying and Designing Clinically Relevant Vision-Language Applications for Radiology
Authors: Nur Yildirim, Hannah Richardson, M. Wetscherek, Junaid Bajwa, Joseph Jacob, M. Pinnock, Stephen Harris, Daniel Coelho De Castro, Shruthi Bannur, Stephanie L. Hyland, Pratik Ghosh, M. Ranjit, Kenza Bouzid, Anton Schwaighofer, Fernando P'erez-Garc'ia, Harshita Sharma, O. Oktay, M. Lungren, Javier Alvarez-Valle, A. Nori, Anja Thieme
Year: 2024


RUNNING...
Stop
Deploy
About This Assistant
This tool acts as a personalized research assistant to students, faculty and researchers
Summarize recent papers
Explore UB faculty research
Find collaboration opportunities
Quick Links
UB Homepage
UB Libraries
Dept. of Computer Science
Feedback
We'd love to hear from you!
Please enter your name
Stephen Harris, Daniel Coelho De Castro, Shruthi Bannur, Stephanie L. Hyland, Pratik Ghosh, M. Ranjit, Kenza Bouzid, Anton Schwaighofer, Fernando P'erez-Garc'ia, Harshita Sharma, O. Oktay, M. Lungren, Javier Alvarez-Valle, A. Nori, Anja Thieme
Year: 2024
View Full Paper
▼ Abstract
Recent advances in AI combine large language models (LLMs) with vision encoders that bring forward unprecedented technical capabilities to leverage for a wide range of healthcare applications. Focusing on the domain of radiology, vision-language models (VLMs) achieve good performance results for tasks such as generating radiology findings based on a patient's medical image, or answering visual questions (e.g., “Where are the nodules in this chest X-ray?”). However, the clinical utility of potential applications of these capabilities is currently underexplored. We engaged in an iterative, multidisciplinary design process to envision clinically relevant VLM interactions, and co-designed four VLM use concepts: Draft Report Generation, Augmented Report Review, Visual Search and Querying, and Patient Imaging History Highlights. We studied these concepts with 13 radiologists and clinicians who assessed the VLM concepts as valuable, yet articulated many design considerations. Reflecting on our findings, we discuss implications for integrating VLM capabilities in radiology, and for healthcare AI more generally.
AI Summary: 1. The study introduces four concepts of vision-language models (VLMs) designed for use in radiology: Draft Report Generation, Augmented Report Review, Visual Search and Querying, and Patient Imaging History Highlights. 2. The researchers found that the VLM concepts were perceived as valuable by 13 radiologists and clinicians, but several design considerations were also highlighted for future development and integration in radiology and healthcare AI more broadly.

References:

1. **Semantic Scholar API Documentation**
   https://api.semanticscholar.org/
   (Used for fetching academic paper metadata like title, abstract, authors, etc.)
2. **arXiv API Documentation**
   https://info.arxiv.org/help/api/index.html
   (Used for accessing open-access science and technology research papers)
3. **Streamlit – Official Docs**
   https://docs.streamlit.io/
   (Used to build the front-end interface of the app)
4. **Ollama – Open Source LLM Serving**
   https://ollama.com/
   (Used for running language models like Mistral locally for summarization tasks)
5. **ChromaDB Documentation**
   https://docs.trychroma.com/
   (Used as a vector database to store and retrieve research embeddings for matching)
6. **RAG (Retrieval-Augmented Generation) Overview**
   Lewis, Patrick, et al. "Retrieval-augmented generation for knowledge-intensive NLP tasks."
   NeurIPS 2020. https://arxiv.org/abs/2005.11401
   (The foundational method your system uses for improved paper recommendations)