# Academic Research Paper Assistant Application

## Problem Statement:

Develop an end-to-end application that functions as an academic research paper assistant using Large Language Models (LLMs). The solution must incorporate frameworks such as Streamlit and FastAPI, and leverage LLMs from Transformers, Ollama, or VLLM. The assistant should effectively assist users in academic research by integrating multi-agentic behaviors involving at least three LLM calls and prompts, along with 1-2 function/API calls. This solution should offer the ability to collect, store, query, summarize, and generate future research directions for a given research topic.

**Requirements**:

1. **Data Collection**: For a given research topic, search for all related research papers and store them in a time-series database.
2. **Capabilities**:
    - **Summarize Research**: Summarize research contributions from the past five years and generate new work ideas based on recent advancements.
    - **Question Answering**: Provide answers to user questions regarding the research papers.
    - **Generate Review Paper**: Generate a review paper summarizing the key points and future research directions.

**Agents to be Implemented**:

1. **Search Agent**: Search the web for research papers related to a given topic, particularly on platforms such as Arxiv.
2. **Database Agent**: Query stored research papers for a specific year based on the given topic.
3. **Q&A Agent**: Handle user questions about a specific paper or a list of given papers, including questions on images, charts, and graphs.
4. **Future Works Agent**: Generate potential improvements and create a review paper suggesting future research opportunities.

**Frontend and User Interaction:**

You need to create a user-friendly interface where users can enter a topic name. The application should then display a list of relevant papers and provide an intuitive chat interface for users. Use a framework like Streamlit or another appropriate front-end technology to enable users to interact through chat, view papers in a timeline format, and select one or more papers for discussion.

**Tools To be used:**

1. Use Transformers, Ollama, or vLLM to create model endpoints. (based on your device compute capability). No calling of closed models like chatGpt, claude or others.Use open models.
2. Use function calling using Outlines or Ollama.
3. For graph db use Neo4j
4. For microservices interaction use FastApi
5. For creating frontend, use Streamlit or any similar for fast creation of frontend.

## What to submit:

1. Screen recorded video of working application.
2. Code repository.

Form link is provided in README file. Please fill that up for submission, once you are done.

**What to Include in the Screen Recording and Code:**

- A functional frontend.
- For each topic, show papers that are added.
- At least two distinct searches for topics, with separate chats for different topics.
- The code repository should include your code as well as a README file explaining your approach.
- Code should be well-structured and appropriately commented.

**Success Criteria and Scoring**:

1. **Basic Level**:
   - **1 Point**: Answer basic questions about a single paper.
   - **1 Point**: Answer questions related to images, charts, and graphs within a paper.
   - **1 Point**: Provide answers for questions about a single paper and show the exact part of the paper where the information was retrieved.
2. **Intermediate Level**:
   - **1 Point**: Answer questions across multiple research papers on a given topic and show from which papers and which part of papers where used to generate that answer.
   - **1 Point**: Summarize findings from multiple papers over a given timeframe.
   - **1 Point**: Extract key information from multiple papers and present it coherently.
3. **Advanced Level**:
   - **1 Point**: Generate ideas for future work to be included in a review paper.
   - **1 Point**: Create a well-structured review paper summarizing future research opportunities.

- **1 Point**: Generate an improvement plan based on key research works, including novel contributions.
- **1 Point**: Combine multiple papers to produce a cohesive improvement plan with suggestions for new research directions.

**Total Points Available**: **10 Points**
**To Pass: 7 Points and above.**

**Sample Example**:
Suppose a user inputs the research topic: "Text-to-SQL."
The application must:

1. **Search Agent**: Fetch related papers on text-to-SQL from platforms such as Arxiv, focusing on papers published in the last five years (e.g., 2019-2024). Store these papers in a time-series database.
2. **Database Agent**: Store the collected papers and allow querying of papers based on specific timeframes, such as retrieving all papers related to text-to-SQL from 2019 onwards.
3. **Frontend Application**: Display all the fetched papers in a user-friendly interface using Streamlit, allowing users to browse the papers easily.
4. **Q&A Agent**: Provide a question-answering feature where users can interact with the research papers. Example questions include:
    - "Summarize the advancements in text-to-SQL models over the last five years."
    - "Explain the table structure presented in Paper Y and its significance."
    - "What does the chart in Paper Z indicate about model accuracy over time?"
    - "Which is the best performing mechanism or model for text-to-SQL, and which paper presents it?"
    - **Example Interaction**:
        - *User*: "Which is the best performing mechanism or model for text-to-SQL?"
        - *Bot*: "Based on the collected papers, Model X presented in Paper C has shown the best performance for text-to-SQL tasks. This model outperforms others in terms of query accuracy and execution time, as demonstrated in benchmark comparisons. Paper C provides detailed analysis, including experimental results, that highlight the superior accuracy and efficiency of Model X."
5. **Future Works Agent**: Suggest potential improvements or research opportunities based on the reviewed papers. Generate a review paper summarizing the state-of-the-art techniques and propose new directions for future work, such as enhancing query accuracy or integrating with newer datasets.

**Chat Interaction Example**:
***User***: "Show me all the papers related to text-to-SQL in the last 5 years."

***Bot***: "Here are the papers on text-to-SQL from 2019 to 2024. [List of papers displayed]"

***User***: "Summarize the advancements made in these papers."

***Bot***: "The advancements in text-to-SQL over the past five years include improved model accuracy, better handling of complex queries, and integration with larger datasets. Notable contributions include Paper A, which introduces a new parsing technique, and Paper B, which enhances the understanding of nested queries."

***User***: "Can you explain the table in Paper B regarding query execution times?"

***Bot***: "The table in Paper B compares query execution times across different models. It shows that Model X has the lowest execution time, indicating a significant performance improvement over previous approach."

***User***: "What future research can be done based on these papers?"

***Bot***: "Based on the reviewed papers, future research could focus on enhancing the adaptability of text-to-SQL models to different database schemas, improving natural language understanding for complex queries, and integrating multimodal data to enrich the query context. A review paper summarizing these directions can also be generated."

***User***: "Which is the best performing mechanism or model for text-to-SQL?"

***Bot***: "According to the papers reviewed, Model X presented in Paper C is the best performing mechanism for text-to-SQL. It has achieved the highest accuracy and fastest query execution times compared to other models. Paper C provides extensive experimental validation, demonstrating its superior performance, especially in handling complex SQL queries."**[Paper 2-section 1.2 , Paper 3-section 2.1]**