Project Overview:

Objective:
Create a Data aggregation platform where users can input their interests (e.g., "latest AI news"). The platform scrapes relevant articles, news, data from web and filters them for relevance using an ML model, feeds them into a VectorDB, which can be used for queries on data collected with Retrieval-Augmented Generation (RAG) model.

Workflow:

1. **User Input:** The user enters a query, e.g., "AI in healthcare."
2. **Web Scraping:** The system scrapes articles from trusted sources.
    - Prompt Template which can identify the website and based on it suggest the sites to be scrapped.
    - Generic and Dedicated scrapers for supporting google and site-specific scrapings.
    - Tools: Selenium and Beautiful soup
3. **Relevance Check:** A pre-trained NLP model filters relevant articles using text embeddings and similarity checks.
    - Embedding Model: OpenAI embedding / Cohere embedding.
    - Retrieval Optimization
        - Base Retriever embedding model finetuning
        - Pre-retrieval optimization: Sentence window retrieval, Query expansion, Sub query generation etc.
        - Retrieval optimization: Hybrid search
        - Post-retrieval optimization: Re-ranking
    - Storing data in vectordb as a new persistent storage directory to differentiate data in between different user queries.
    - Tools: ChromaDB
4. **User Dashboard:** The summarized content, alongside an interactive dashboard, allows users to explore the content visually.
    - Basic visualizations about scraped data with some text visualizations like word cloud.
5. **Summarization:** Relevant articles are summarized into shorter versions using an LLM model and displayed to the user.
    - Use the LLM to summarize the content and provide a short summary of the data scraped.

6. **RAG:** The system retrieves the data collected for new user queries and generate responses.
   - Pass the embeddings from relevance module and use the same embedding model and LLM used earlier in relevance module.

Key Components:

1. **Scraping Module**
   - **Purpose:** Scrape relevant articles, news, data based on user input.
   - **Tools:** BeautifulSoup, Scrapy, or Selenium (for dynamic content).
   - **Steps:**
     - Accept user queries like "AI advancements" or "Cryptocurrency updates."
     - Perform web scraping to gather news articles from various sources.
     - Store the scraped data in a database for further analysis.
2. **Relevance Filtering Module**
   - **Purpose:** Filter and classify the scraped data to determine relevance to the user query.
   - **Tools:**
     - ML Models: Use pre-trained NLP models like BERT, T5, or fine-tuned transformer models to assess the semantic relevance of articles to the user's input.
     - Vector Embeddings: Use sentence embeddings (e.g., with Hugging Face Transformers) to match user intent with article content.
     - Similarity Search: Use libraries like Faiss to quickly compare and find relevant articles from the scraped data.
3. **Dashboard and Analytics Module**
   - **Purpose:** Present the results in a user-friendly dashboard.
   - **Tools:**
     - Visualization Tools: Streamlit, Dash, or Gradio for building interactive user dashboards.
     - Backend Framework: FastAPI or Flask to handle user requests, news scraping, and result display.

        - Database: Use PostgreSQL or MongoDB for storing user queries, scraped articles, and summaries.

4. **Summarization Module**
    - **Purpose:** Summarize relevant articles for quick consumption.
    - **Tools:**
        - LLM models: GPT-3, OpenAI, or Mistral for summarization tasks.
        - Summarization Techniques: Use abstractive or extractive summarization techniques based on article length and user preference.
        - Fine-tune models on articles to optimize performance for summarization.
5. **Retrieval-Augmented Generation (RAG) Module**
    - **Purpose:** Provide enhanced and optimized answers for user queries by combining relevant articles with a generative AI model.
    - **Tools:**
        - VectorDB: As the vector database for retrieving relevant articles.
        - LLM (e.g., Mistral or GPT): Fine-tune the model to summarize or generate answers based on the retrieved information.
        - LangChain: For orchestration between the retrieval and generative parts.

6. **MLOps**
    - **Purpose:** To monitor, scale, and manage the model lifecycle.
    - **Tools:**
        - Model Monitoring & Management: MLflow or Weights & Biases for experiment tracking and model performance monitoring.
        - Data and LLM monitoring: Evidently for Data Drift, Model Drift, LLM drift monitoring.
        - Scheduling: Airflow or Prefect for automating the news scraping and data processing pipelines.
        - Grafana and Prometheus for Endpoints logging and monitoring
        - Containerization: For running scheduled flows for ML monitoring, retraining or data collection.

Prerequisites:

- Python
- Web Scraping
- LLMs (OpenAI, HuggingFace, Ollama / llama.cpp)
- VectorDB

Tools:

- Selenium/Soup
- Transformers
- Opensource VectorDB: ChromaDB, Faiss, Qdrant / Cloud VectorDB: Pinecone
- LLM for RAG (OpenAI, or Mistral / Llama if the opensource model needs to be used)
- Streamlit/ Gradio for basic UI
- FastAPI
- Relational DB: MySQL, Postgres / Cloud Postgres type DB: Supabase
- MLflow
- Evidently AI
- Airflow / Prefect
- Docker
- Grafana & Prometheus