* **Technology:** Python, Streamlit, Selenium, WebDriver Manager, Pandas, NumPy, Google Generative AI (for Gemini API), FAISS ,LangChain
* **Algorithms & Concepts:** Web Scraping, DOM Parsing (CSS Selectors, XPath), (LLMs), Prompt Engineering, **Retrieval-Augmented Generation (RAG)**, Text Splitting & Chunking, Vector Embeddings (embedding-001), Vector Database (FAISS Index), K-Nearest Neighbors (KNN) Similarity Search (L2 Distance), API Integration.
* **Methodology:**
  1. **User Input:** The process begins when the user provides a Flipkart product URL.
  2. **Data Extraction:** A Selenium-powered scraper launches a headless Chrome browser, navigates to the reviews page, and extracts the raw text from multiple pages of customer reviews.
  3. **AI Summarization:** All scraped reviews are concatenated and sent to the Gemini API with a specifically engineered prompt to generate a concise pros and cons summary.
  4. **RAG System Setup (Chatbot Indexing):**
     + **Chunking:** The reviews are broken into small, overlapping chunks using LangChain.
     + **Embedding:** Each chunk is converted into a numerical vector representation using Google's embedding model.
     + **Indexing:** All vectors are loaded into an in-memory FAISS index, creating a searchable knowledge base.
  5. **Interactive Q&A (RAG Retrieval & Generation Loop):**
     + A user submits a question through the Streamlit interface.
     + The question is converted into a vector.
     + FAISS performs a similarity search on the index to retrieve the most relevant review chunks.
     + These retrieved chunks (the context) are combined with the original question into a new prompt.
     + The Gemini API receives this final prompt and generates a factually-grounded answer.
  6. **Presentation:** The generated summary and the interactive chatbot are displayed to the user in a clean web interface built with Streamlit.

**Technology:** Python, Streamlit, Selenium, WebDriver Manager, Pandas, NumPy, Google Generative AI (for Gemini API), FAISS ,LangChain

**Algorithms & Concepts:** Web Scraping, DOM Parsing (CSS Selectors, XPath), (LLMs), Prompt Engineering, **Retrieval-Augmented Generation (RAG)**, Text Splitting & Chunking, Vector Embeddings (embedding-001), Vector Database (FAISS Index), K-Nearest Neighbors (KNN) Similarity Search (L2 Distance), API Integration.

**Methodology: User Input->Data Extraction->AI Summarization->RAG System Setup (Chatbot Indexing)->Interactive Q&A (RAG Retrieval & Generation Loop):**

Of course. Here are the direct links to the key resources mentioned in the literature review, which you can cite in your project documentation.

**Key Literature & Resources**

1. **Retrieval-Augmented Generation (RAG)**
   * **Title:** **Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks**
   * **Relevance:** This is the foundational academic paper that introduced the RAG architecture. Citing this shows you understand the theoretical basis of your chatbot.
   * **Link:** <https://arxiv.org/abs/2005.11401>
2. **Google Gemini Models**
   * **Title:** **Gemini: A Family of Capable Multimodal Models**
   * **Relevance:** This is the official technical report from Google detailing the models you are using (Gemini 1.0/1.5). It's the primary source for the core AI engine.
   * **Link:** <https://arxiv.org/abs/2312.11805>
3. **FAISS (Vector Search)**
   * **Title:** **Billion-Scale Similarity Search with GPUs**
   * **Relevance:** This paper details the highly efficient algorithms behind FAISS, the library you use for finding relevant reviews. It's a great citation for the "vector search" component.
   * **Link:** <https://arxiv.org/abs/1702.08734>
   * **Alternative (Practical) Citation:** For a software tool, citing the official GitHub repository is also common.
   * **Link:** <https://github.com/facebookresearch/faiss>
4. **Selenium (Web Scraping)**
   * **Title:** **The Selenium Project Official Documentation**
   * **Relevance:** For established software tools like Selenium, the official documentation is the best and most appropriate source to cite. It shows you're referencing the tool's standard implementation.
   * **Link:** <https://www.selenium.dev/documentation/>

**The Diagnosis: Why It Stopped**

Your scraper successfully fetched 30 pages of reviews before it stopped. This happens for two main reasons on commercial websites:

1. **Pagination Change (Most Likely):** After a certain number of pages (like 20-30), websites like Flipkart often stop showing a simple "Next" button. Instead, they switch to a numbered page system (e.g., < 1 ... 30, 31, 32 ... >). Your current scraper is designed to look only for the "Next" button, so when it disappears, the scraper correctly determines it has reached the end and stops.
2. **Rate Limiting:** After dozens of rapid page requests from the same computer, a website's server will often temporarily throttle or block the connection to prevent being overloaded by bots. This is a standard defense mechanism.