Smart Assistant for Research Summarization

(AI ASSISTANT : USING HUGGING FACE TRANSFORMERS)

Submitted By : Taniya Dixit, DTC **Submission date : 05.07.2025**

Abstract

"This project presents a multi-modal research summarization assistant that supports OpenAI APIs, locally hosted LLMs via Ollama, and offline open-source transformer models to generate summaries, answer questions, and challenge user comprehension based on uploaded documents."

Introduction

In the digital age, researchers, students, and professionals deal with an overwhelming volume of academic papers, reports, and technical documents. Understanding and extracting valuable insights from these texts requires time, effort, and attention to detail. With the advancement of artificial intelligence and natural language processing (NLP), it has become possible to build systems that can read, summarize, and answer questions from documents with high accuracy.

This project, "Smart AI Research Summarization Assistant", aims to create an intelligent system that can read research papers both (PDF/TXT), summarize their content, answer user queries, and evaluate comprehension. It supports three implementation modes—OpenAI API, Ollama-based local LLMs, and a fully offline Hugging Face Transformers setup—to adapt to various environments and resource constraints.

Purpose of the Project

The purpose of this project is to:

1. Assist users in quickly understanding long and technical research documents.
2. Provide instant summaries to reduce manual reading effort.
3. Enable intelligent question-answering for focused information retrieval.
4. Offer a challenge mode to test users' comprehension with logic-based questions.
5. Support flexible deployment through API-based, local LLM, and offline implementations.
6. Promote open-source accessibility by ensuring one mode works without internet or paid APIs.

Need for Automated Summarization and Comprehension Tools

The modern research landscape is saturated with new publications every day across domains like medicine, machine learning, law, and engineering. Professionals and students struggle to keep up with this volume due to:

- Time constraints
- Complex technical language
- Lengthy and dense formats

Manually reading each document for summaries and insights is inefficient. There is a strong need for **AI-powered summarization and comprehension tools** that can:

- Automatically extract the core meaning of documents.
- Generate accurate summaries in seconds.
- Support natural question-answering from unstructured content.
- Run efficiently even in offline or resource-constrained environments.

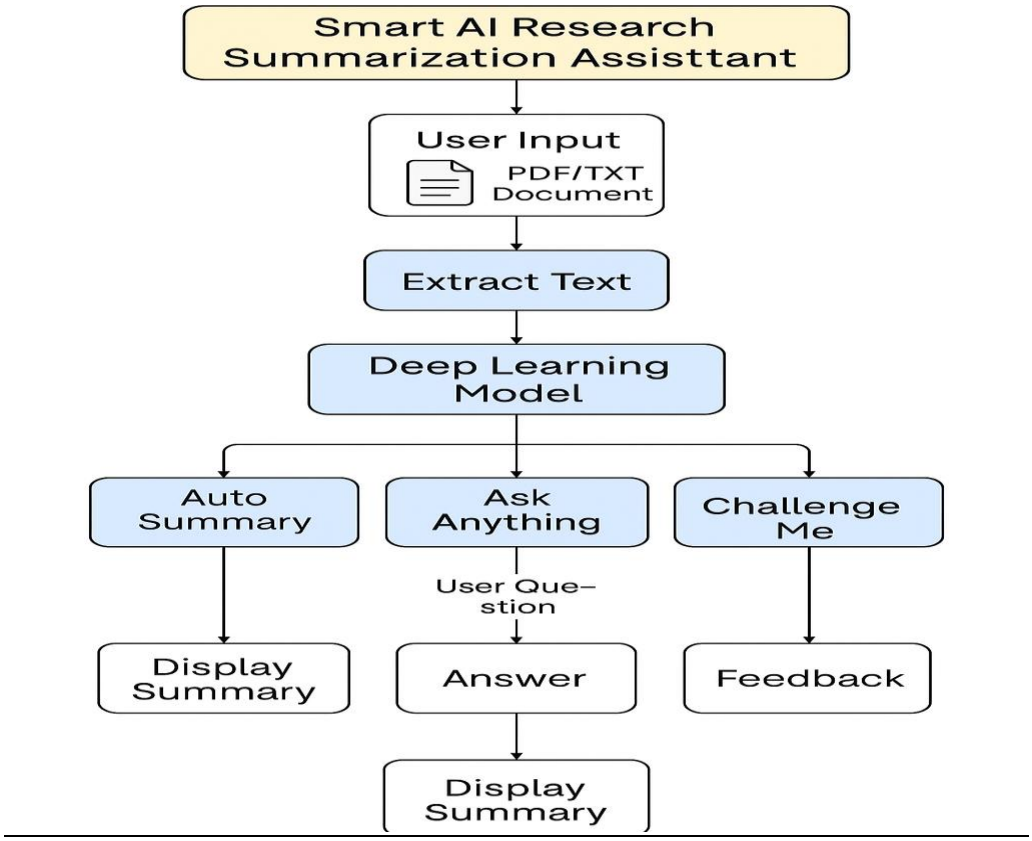
Problem Statement

Reading and understanding research documents is time-consuming and challenging, especially for non-experts or those under time pressure. Existing solutions for summarization and Q&A either rely on internet connectivity (OpenAI), require significant computing resources (local LLMs), or lack versatility (single-purpose summarizers).

Hence, the problem can be stated as:

"How can we design a smart assistant that can automatically summarize research documents, support context-aware question answering, and evaluate user understanding—while offering flexibility through cloud APIs, local models, and fully offline operation?"

FLOWCHART :



EXPLAINED PROCEDURE

Step-by-Step: Components Breakdown:

✅ 1. File Upload & Text Extraction

Purpose: Accepts user documents in PDF or TXT format and converts them into plain text for processing.

How it works:

- Uses Streamlit for the upload interface.
- If PDF: pdfplumber is used to extract text page by page.

2. LLM Interface (OpenAI / Ollama / Transformers)

Purpose: Connects the extracted text to a language model (LLM) to generate summaries, questions, or answers.

Function Types:

- generate_summary(text)
- generate_questions(text)
- evaluate_answer(question, user_answer, text)

✅ 3. Q&A Pipeline

Purpose: Allows the user to ask a question based on the uploaded content and receive a relevant, accurate answer.

How it works:

- For OpenAI/Ollama: passes the question and text to the LLM.
- For Offline Mode:
  - Uses FAISS to find the most relevant paragraphs.
  - Passes top 2–3 chunks to a lightweight question-answering model (e.g. distilbert-squad).

✅ 4. Vector Search (Only for Offline Mode)

Purpose: Finds the most relevant parts of the document for Q&A and comprehension using semantic similarity.

How it works:

- Text is split into smaller chunks (≈ 100–500 words).
- Each chunk is encoded into a vector using sentence-transformers.
- Vectors are indexed using faiss.IndexFlatL2.
- At query time, the question is embedded and searched for top-k matching chunkS.
-

IMPLEMENTATION DETAILS

This assistant is designed to be modular and can run in three different modes depending on the user's system resources and internet availability:

a. OpenAI Mode

- **Backend Used**: OpenAI GPT models via API
- **Library**: openai.ChatCompletion
- **Required**:
  - OpenAI API key
  - Internet connection
- **Models Supported**: gpt-3.5-turbo, gpt-4

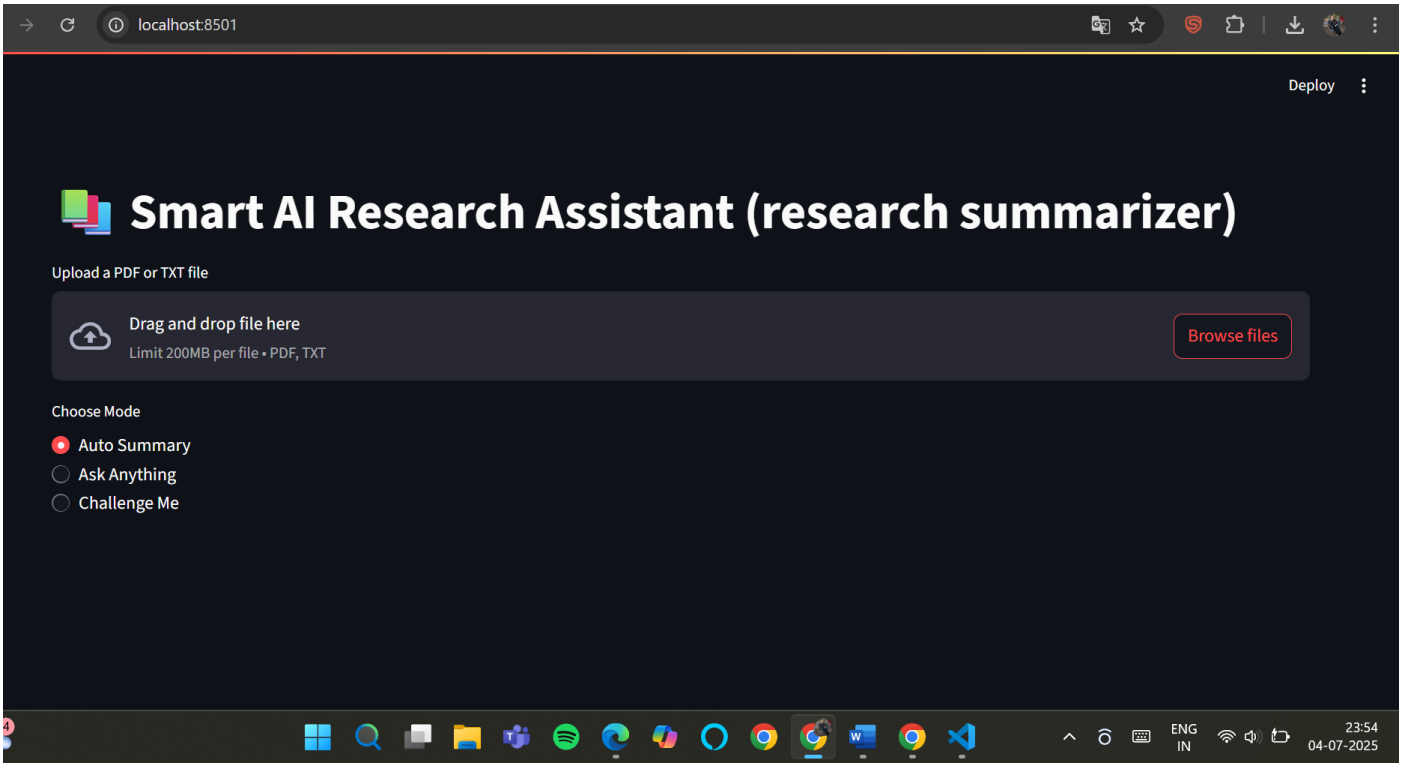
b. Ollama Mode

- **Backend Used**: Local models served by Ollama
- **Library**: langchain.llms.Ollama
- **Required**:
  - Ollama installed locally
  - Sufficient RAM (4GB–6GB+ depending on model)
- **Models Supported**: mistral, phi3, llama3, gemma, etc.

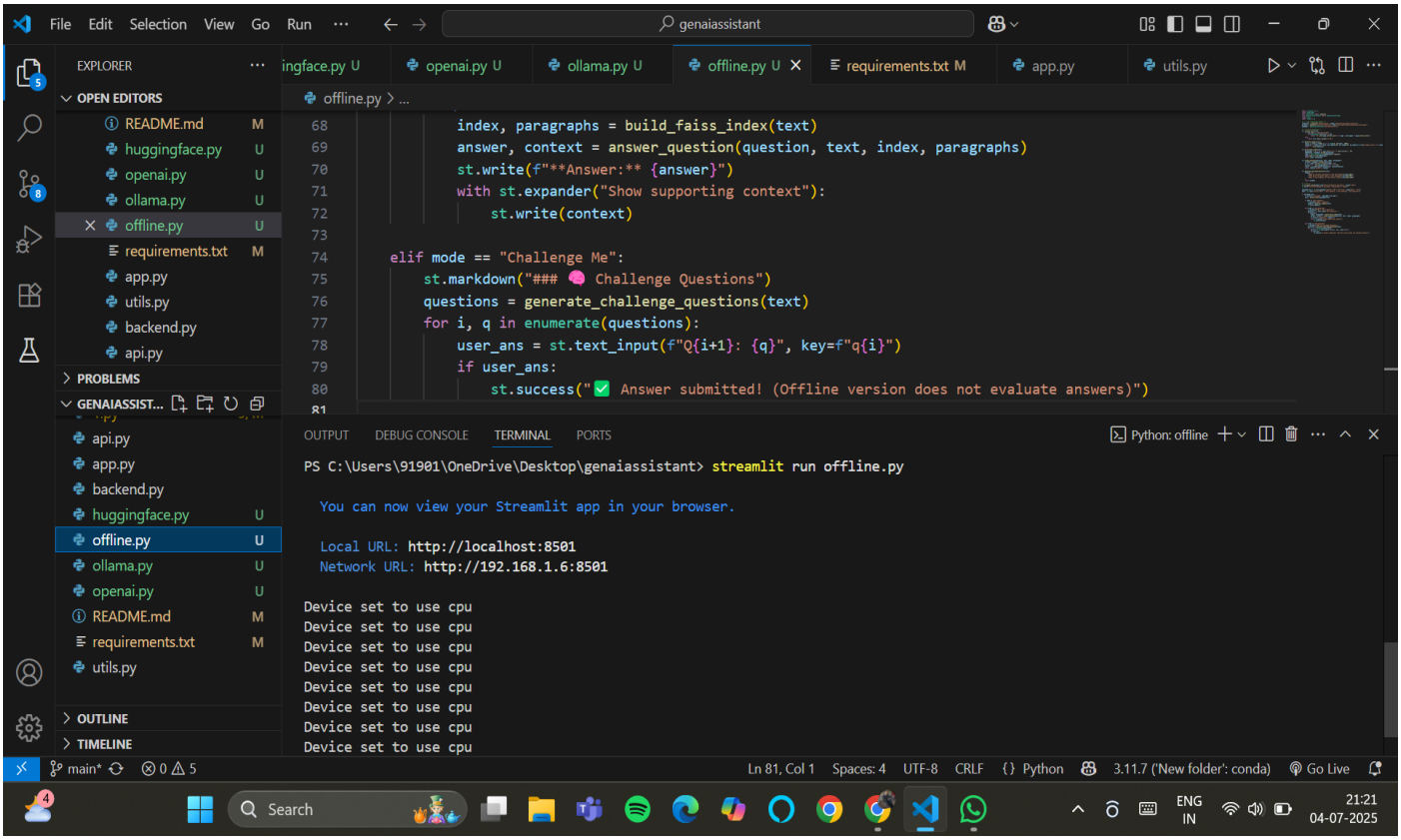
c. Offline Mode (Transformers)

- **Backend Used**: Hugging Face Transformers + FAISS
- **Libraries**:
  - transformers (for summarization and QA)
  - sentence-transformers (for embeddings)
  - faiss-cpu (for vector search)
- **Models Used**:
  - sshleifer/distilbart-cnn-12-6 for summarization
  - distilbert-base-uncased-distilled-squad for Q&A
  - all-MiniLM-L6-v2 for semantic similarity
- **Required**:
  - Python environment (≥3.7)
  - No internet after initial model download

USER INTERFACE :



CODE :

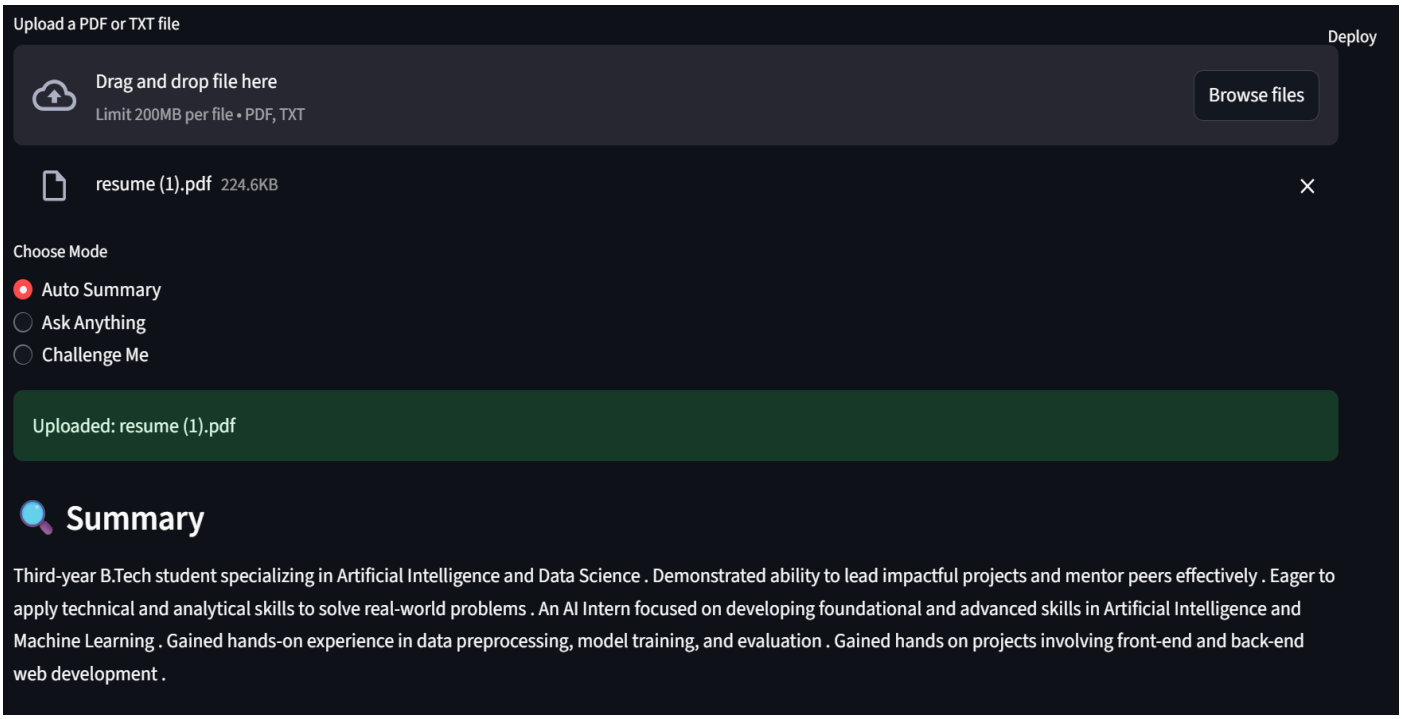


```
            index, paragraphs = build_faiss_index(text)
            answer, context = answer_question(question, text, index, paragraphs)
            st.write(f"**Answer:** {answer}")
            with st.expander("Show supporting context"):
                st.write(context)

    elif mode == "Challenge Me":
        st.markdown("### 🧠 Challenge Questions")
        questions = generate_challenge_questions(text)
        for i, q in enumerate(questions):
            user_ans = st.text_input(f"Q{i+1}: {q}", key=f"q{i}")
            if user_ans:
                st.success("✅ Answer submitted! (Offline version does not evaluate answers)")
```

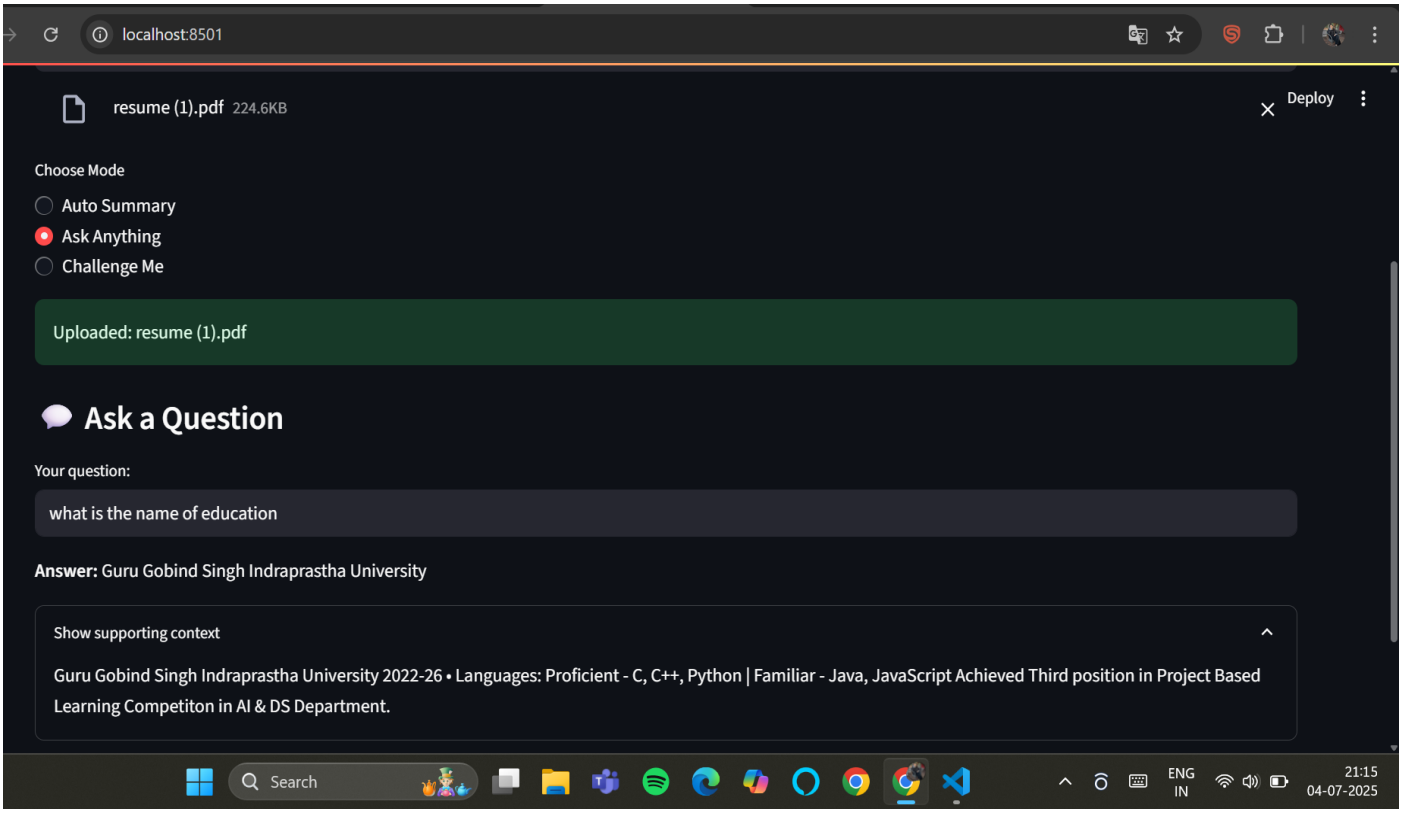
```
PS C:\Users\91901\OneDrive\Desktop\genaiassistant> streamlit run offline.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.1.6:8501

Device set to use cpu
Device set to use cpu
Device set to use cpu
Device set to use cpu
Device set to use cpu
Device set to use cpu
Device set to use cpu
Device set to use cpu
```

OUTPUT :

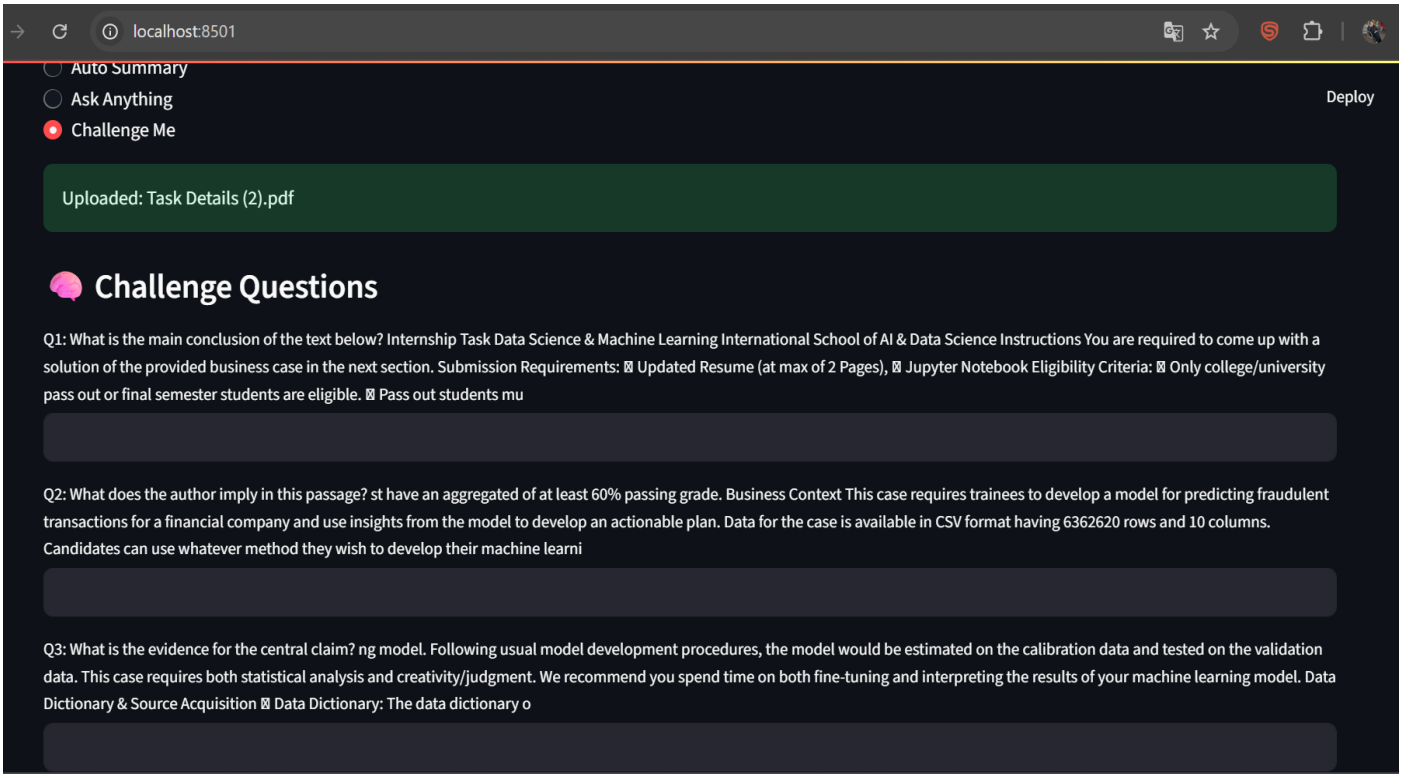
a) **Generate summary**



b) **Ask Anything :**



c) **Challenge me :**



CONCLUSION:

The Smart AI Research Summarization Assistant project successfully demonstrates how artificial intelligence can assist users in efficiently understanding and interacting with research documents. By integrating document summarization, question answering, and logic-based comprehension evaluation, the system provides an intelligent interface that adapts to different technical and resource constraints.

The project was implemented in three distinct modes:

- OpenAI Mode, for high-accuracy results using GPT models in cloud environments.
- Ollama Mode, for fully local LLM execution on capable machines.
- Offline Transformers Mode, for lightweight summarization and Q&A without any internet dependency.

Each mode serves a specific user segment—ranging from cloud-powered environments to privacy-sensitive or offline setups—making the assistant versatile and accessible. The modular design ensures the system can be further extended to support other models, languages, or additional tasks such as keyword extraction, translation, or speech-to-text.

In summary,

This assistant bridges the gap between dense technical documents and user-friendly insights, leveraging modern NLP advancements to promote faster learning, efficient research, and enhanced digital comprehension.