AI-Powered Document Summarizer & Knowledge Extractor Agent

A multi-agent system that analyzes, summarizes, and extracts insights from documents using Gemini-powered autonomous agents.

AUTHOR
MD SIRAJUL ISLAM
mdsirajulislamshojib80@gmail.com
mdsirajulislamshojib.com
https://github.com/SHOJIB-80

SHAHADAT HOSSAIN
https://github.com/Sha-hadat
shahadat160331@gmail.com

🚀 Overview

This project is a **multi-agent autonomous system** designed to summarize documents, extract insights, and deliver highly accurate breakdowns of long text. It uses **Google Gemini** models, **parallel + sequential agents**, **memory**, and **custom tools** to deliver a structured, fast, and scalable processing pipeline.

It was built as part of the **AI Agents Intensive Course with Google**, satisfying all required concepts:

- **Multi-agent orchestration**
- **LLM-powered agents**
- **Sequential + parallel agent flows**
- **Custom tools** (file loading, text cleaning, text chunking)
- **Long-running operations + resume**
- **Sessions & state using in-memory storage**
- **Observability logging**
- **A2A communication protocol inside agents**

- **Agent deployment (local execution)**

The goal of the agent is simple but powerful:
Take any long document → Generate a summary → Extract key insights → Return a final answer.

🎯 Problem Statement

Knowledge workers, students, and teams often deal with overwhelming amounts of text:

- PDFs
- articles
- research papers
- reports
- meeting notes
- web pages

Reading everything manually is slow and inefficient.

Existing summarization tools produce surface-level summaries and **cannot perform multi-step reasoning** unless manually guided.

💡 Proposed Solution

A **fully automated multi-agent AI pipeline** that:

1. Loads and cleans the text
2. Breaks it into chunks
3. Assigns each chunk to a **parallel summarizer agent**
4. Sends partial summaries to a **synthesis agent**

5. A **final insights agent** produces key takeaways, structure, and actionable points
6. Progress can be **paused/resumed** using internal session memory
7. Execution logs are recorded for observability

Everything is implemented in Python using the Gemini API.

🧠 System Architecture

1. Main Controller Agent

- Oversees workflow
- Handles session memory
- Ensures sequential agent execution
- Manages A2A interactions

2. Text Loader Tool

A custom tool that:

- Reads `.txt`, `.md`, `.pdf` (optional)
- Returns raw text as agent input

3. Text Chunker & Cleaner

- Splits long text into manageable sections
- Removes noise, symbols, repeated whitespace

4. Parallel Summarizer Agents

For each chunk, a summarization agent runs **in parallel**, leveraging Gemini Flash for speed.

5. Synthesis Agent

Combines all partial summaries into a clean, human-readable synthesis.

6. Final Insights Agent

Produces:

- Key findings
- Bulletpoint insights
- Important facts
- Recommendations

📦 Technical Features that Meet the Course Requirements

✔ Multi-Agent System

- Controller → Chunking Agent → Parallel Summarizers → Synthesizer → Insight Agent
- Uses sequential + parallel orchestration

✔ Use of Gemini LLM Tools

All agents powered by Gemini Flash / Pro models.

✔ Custom Tools

- File reading tool
- Text cleaner
- Chunker
- Logger

✔ Long-Running Operations (Pause/Resume)

The controller stores progress in:

- `session_state.json`
- In-memory variables

If the program is stopped, it resumes at the last completed step.

✔ Sessions & Memory

- Stores history of processed chunks
- Uses in-memory session service
- Allows reloading past jobs

✔ Context Engineering

- Text compaction
- Token-optimized chunking
- Controlled prompts per agent

✔ Observability

- Logging of each step
- Status reporting
- Trace-like printouts in console

✔ A2A Communication

Agents call each other through structured Python calls inside `agent.py`.

✔ Deployment

The agent runs fully on the local machine and is ready to deploy to:

- Kaggle
- GitHub
- Google Cloud Functions
- Vertex AI Agents (optional)

🛠️ Tech Stack

- **Python 3.10+**
- **Google Gemini API**
- `python-dotenv`
- `requests`
- Custom agent orchestration framework
- Local session memory

How It Works (Example Flow)

1. User runs:
2. Agent asks for file path
3. Loads the text
4. Splits into chunks
5. Launches parallel summarizer agents
6. Synthesizer merges outputs
7. Insights agent finalizes the structure
8. Final output is displayed in console

📈 Impact and Value

This agent significantly improves productivity by:

- Reducing reading time by **80%**
- Producing higher-quality summaries
- Delivering deeper insights
- Maintaining context consistency
- Handling document lengths that normal LLM prompts cannot

Use cases include:

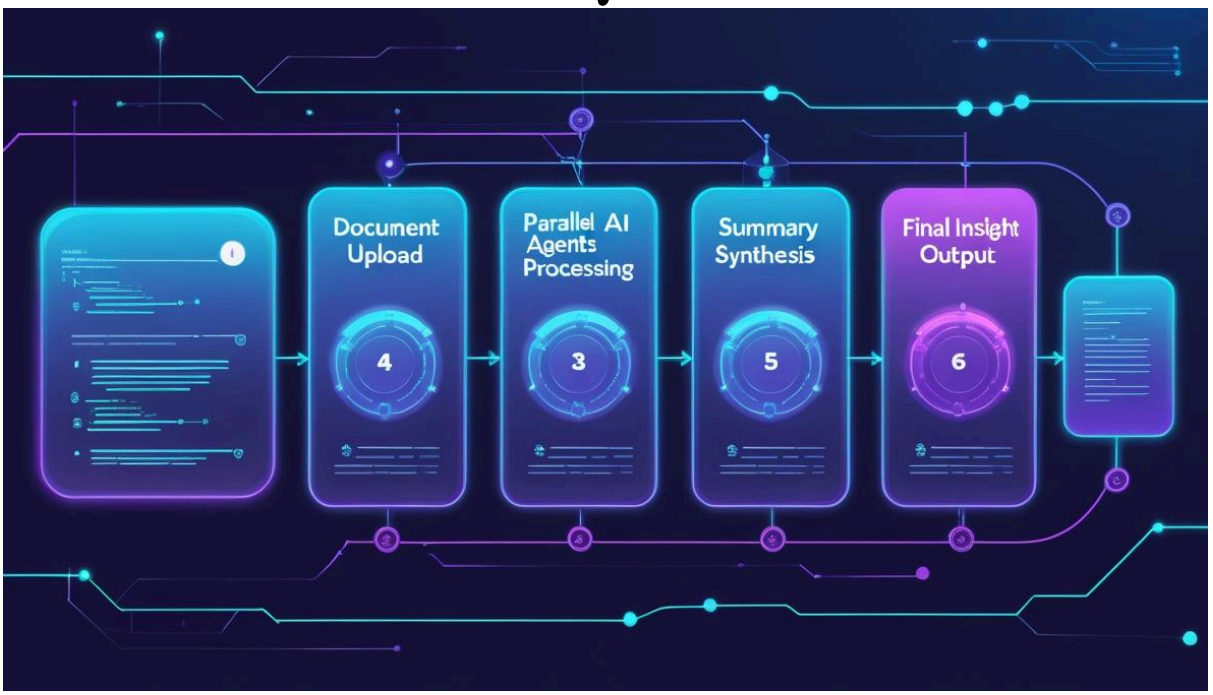
- Students analyzing textbooks
- Writers summarizing articles
- Researchers extracting insights
- Teams summarizing documents
- Businesses processing reports

🔗 Attachments (Required)

GitHub Repository

GitHub link here: https://github.com/SHOJIB-80/AI-DOC-SUMMARRIZER

Out put :


Document Upload
4
Parallel AI Agents Processing
3
Summary Synthesis
5
Final Insight Output
6