

## Task Title:

### Agentic RAG Chatbot for Multi-Format Document QA using Model Context Protocol (MCP)

---






## Problem Statement:

You are required to build an **agent-based Retrieval-Augmented Generation (RAG) chatbot** that can answer user questions using uploaded documents of various formats. Your architecture must follow an **agentic structure** and should incorporate **Model Context Protocol (MCP)** as the mechanism for communication between agents and/or agents ↔ LLMs.

---

## Core Functional Requirements

Your solution must:

1. **Support Uploading & Parsing of Diverse Document Formats:**
  -  PDF
  -  PPTX
  -  CSV
  -  DOCX
  -  TXT / Markdown
2. **Agentic Architecture (minimum 3 agents):**
  - IngestionAgent: Parses & preprocesses documents.
  - RetrievalAgent: Handles embedding + semantic retrieval.
  - LLMResponseAgent: Forms final LLM query using retrieved context and generates answer.





3. **Use Model Context Protocol (MCP):**
  - Each agent must send/receive messages using **structured MCP-like context objects**, such as:

```
{
  "sender": "RetrievalAgent",
  "receiver": "LLMResponseAgent",
  "type": "CONTEXT_RESPONSE",
  "trace_id": "abc-123",
  "payload": {
    "top_chunks": ["...", "..."],
    "query": "What are the KPIs?"
  }
}
```

- You can implement MCP using in-memory messaging, REST, or pub/sub.
- 4. **Vector Store + Embeddings**
  - Use any embeddings (OpenAI, HuggingFace, etc.)
  - Use a vector DB (FAISS, Chroma, etc.)
- 5. **Chatbot Interface (UI)**
  - Allow users to:
    - Upload documents
    - Ask multi-turn questions
    - View responses with source context
  - Use any UI framework: Streamlit, React, Angular, Flask, etc.

---

## Deliverables

1.  **GitHub Repository**
  - Include:
    - Well-organized code
    - Clear README.md with setup instructions
2.  **PPT Presentation**
  - Slide deck (3–6 slides) must include:
    - Agent-based architecture with MCP integration
    - System flow diagram (with message passing)
    - Tech stack used
    -  UI screenshots of working app
    - Challenges Faced while doing the project
    - (Optional) future scope / improvements
3.  **Submission**
  - Share:
    - Public GitHub repository link
    - Architecture PPT (PDF or PPTX) [ To be included in the GitRepo Itself]
    - Include a Video for 5 mins where 1 min give the application demo, 2 min architecture and flow explanation, 2 min code explanation. (Its optional to show face)

---

## Sample Workflow (Message Passing with MCP)

User uploads: sales\_review.pdf, metrics.csv

User: "What KPIs were tracked in Q1?"

- ➡ UI forwards to CoordinatorAgent
- ➡ Coordinator triggers:
  - ◆ IngestionAgent → parses documents
  - ◆ RetrievalAgent → finds relevant chunks
  - ◆ LLMResponseAgent → formats prompt & calls LLM
- ➡ Chatbot shows answer + source chunks

MCP message example:

```
{
  "type": "RETRIEVAL_RESULT",
  "sender": "RetrievalAgent",
  "receiver": "LLMResponseAgent",
  "trace_id": "rag-457",
  "payload": {
    "retrieved_context": ["slide 3: revenue up", "doc: Q1
summary..."],
    "query": "What KPIs were tracked in Q1?"
  }
}
```