💡 Hackathon Task Division – 5 Team Members

👩‍💻 1. Google Drive Integration & File Extraction (API Integrator)

Goal: Get access to files, extract their text and metadata

Tasks:

- Set up **Google Drive API** with a **service account**
- Create script to:
    - List files from the dummy Drive
    - Download files (PDF, DOCX, TXT)
    - Extract content:
        - Use `pdfplumber` or `PyMuPDF` for PDFs
        - Use `python-docx` for Word files
        - (Optional) Use `pytesseract` for image OCR
- Structure extracted content into JSON format (filename, content, metadata)

Tools:

- Python, `google-api-python-client`, `pdfplumber`, `python-docx`

👨‍🔬 2. Embedding & Vector Store (ML Integrator, System Designer)

Goal: Convert file content into embeddings and store them for search

Tasks:

- Use **Vertex AI's `textembedding-gecko` model** to generate embeddings

- Split extracted content into chunks (e.g., 300–500 tokens)
- Store embeddings in **FAISS** vector DB (or **ChromaDB** as fallback)
- Write utility to update the vector store when new files are added

Tools:

- Python, `google-cloud-aiplatform`, `faiss-cpu`, `langchain`

🧠 3. AI Agent & Question Answering Logic (RAG Architect)

Goal: Use Gemini Pro to answer questions based on retrieved document chunks

Tasks:

- Set up **Gemini Pro via Vertex AI** for answering queries
- Write the **RAG logic**:
  - Take user question
  - Retrieve top-k chunks from FAISS based on similarity
  - Pass chunks + question to Gemini
  - Return formatted answer
- Optimize for clarity, relevance, and completeness

Tools:

- Python, `google-cloud-aiplatform`, `langchain`, `tiktoken` (for chunking)

👩‍🎨 4. File Categorization & Reorganization (Automation Specialist) - Isaac

Goal: Automatically tag and organize files into Drive folders

Tasks:

- Define logic (or LLM prompt) to classify files into categories:
  - e.g., “If document mentions donors + events → Development”
- Use Drive API to:
  - Add custom labels or tags (if supported)
  - Move files into organized folder structures
    - E.g., `/Marketing/2023_Q4`
- Integrate this script to run after embedding step

Tools:

- Python, `google-api-python-client`, optional: `gemini-pro` classification prompts

🧑‍💻 5. Frontend or CLI Interface (UX/Tooling Developer) - Raylene

Goal: Build a user interface for testing the AI agent

Option A – Web UI (if time allows):

- Use **Streamlit** or **Flask**
- Features:
  - Upload files
  - Ask questions
  - View AI answers
  - Trigger reorganization

Option B – Command-Line Tool (quicker):

- Create a CLI interface to:
    - Upload a document
    - Ask a question
    - See the answer
    - Reorganize Drive content

Tools:

- `streamlit`, `flask`, `typer`, `argparse`, `requests`

📦 Suggested Folder Structure (Shared Git Repo)

bash
CopyEdit

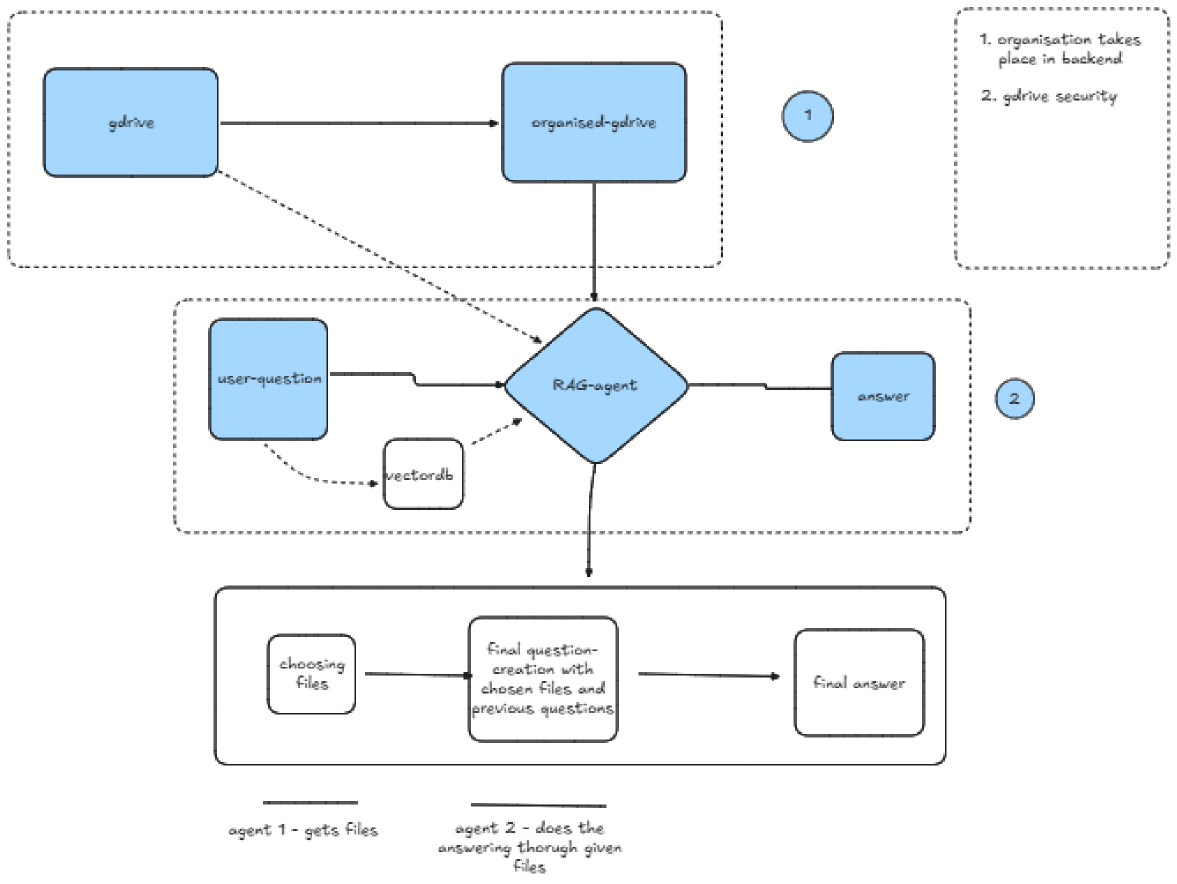
```
ai-agent/
├── drive_integration/        # Member 1
│   └── download_files.py
├── embedding_engine/         # Member 2
│   └── embed_text.py
├── question_answering/       # Member 3
│   └── rag_agent.py
├── file_organizer/           # Member 4
│   └── categorize_files.py
├── interface/                # Member 5
│   └── streamlit_app.py or cli_tool.py
├── vector_store/             # FAISS index & helpers
├── config/
│   └── credentials.json
├── .env
└── README.md
```

✅ Tip: Assign one person to handle integration in the final hours

Have one person (maybe Member 3 or 4) handle:

- Connecting all modules
- Managing credentials and `.env` files
- Ensuring clean output/loggin

Program Flow


gdrive
organised-gdrive
1
1. organisation takes place in backend
2. gdrive security
user-question
RAG-agent
answer
2
vectordb
choosing files
final question-creation with chosen files and previous questions
final answer
agent 1 - gets files
agent 2 - does the answering thorugh given files