AI Transformation Specialist – Technical Assignment

Duration: 2–3 Days
Role: AI Transformation Specialist (2–4 Years Experience)
Company: Techculture.ai

Overview

This assignment evaluates your practical ability to design and implement an end-to-end AI solution using Retrieval-Augmented Generation (RAG) and agentic AI concepts. You will build a small, functional AI assistant that demonstrates your understanding of GenAI, vector databases, LLM orchestration, and business-focused solution design.

Problem Statement

You are tasked with creating a **Service Information Assistant** for a technology services company. This assistant will:

- Ingest and index company service documents (AI consulting, digital marketing, web/app development, FAQs, case studies)
- Answer user questions accurately using a RAG pipeline
- Demonstrate basic "agentic" behaviour by calling a simple tool or API when appropriate
- Provide source citations for transparency and trust

Scenario: A potential client or internal team member asks questions like:

- "What AI services do you offer?"
- "Do you have experience with fintech projects?"
- "What's the estimated cost for a 3-month digital marketing campaign?"

Your assistant should retrieve relevant information from documents and generate accurate, context-aware answers.

Core Requirements

1. Data Preparation & Document Ingestion

- Create or use 5–10 text sources (Markdown, TXT, or PDF) representing:
  - Service descriptions (AI consulting, digital marketing, development)
  - FAQ section
  - 1–2 case study examples or client scenarios
- If we don't provide real documents, create dummy content that simulates realistic service information
- Design the system to be scalable for real production documents

2. RAG Pipeline Implementation

Build a complete Retrieval-Augmented Generation pipeline:

- **Document Processing**: Split documents into semantic chunks (appropriate size for retrieval)
- **Embeddings**: Generate vector embeddings using a suitable model (e.g., OpenAI embeddings, sentence-transformers)
- **Vector Storage**: Store embeddings in a vector database (FAISS, Chroma, Pinecone, or similar)
- **Retrieval**: Implement semantic search to retrieve top-k relevant chunks for user queries
- **Generation**: Use an LLM (OpenAI API, Hugging Face, or other) to generate answers grounded in retrieved context

Key Feature: All answers must include **source citations** showing which document/section was used (e.g., "Source: services_overview.txt, Section: AI Consulting").

3. Agentic Behaviour (Tool Integration)

Implement at least **one tool** that your AI assistant can call autonomously:

Option A – Sentiment Analyzer:

- Classify user query sentiment (Positive/Negative/Neutral)
- Include sentiment context in the response when relevant

Option B – Pricing Estimator:

- Simple Python function that takes service type and duration
- Returns rough cost estimate range
- Agent decides when to call this based on query intent

Option C – Custom Tool:

- Design your own lightweight tool relevant to the business context
- Demonstrate autonomous decision-making about when to use it

You may use LangChain agents/tools framework or implement your own orchestration logic.

4. User Interface

Provide at least one of the following:

Option A – REST API (FastAPI):

- Endpoint: `POST /ask`
- Request body: `{"question": "your question here"}`
- Response: Answer + sources + tool outputs (if applicable)
- Include basic error handling

Option B – Interactive Interface:

- Simple Streamlit app or CLI interface
- User enters question and sees answer with sources
- Display tool usage when triggered

Technical Expectations

Required Stack

- **Language:** Python 3.8+
- **Core Libraries (suggested):**
  - LangChain or similar for RAG/agent orchestration
  - FAISS / Chroma / Pinecone for vector storage
  - OpenAI API / Hugging Face / other LLM
  - FastAPI or Streamlit (for interface)
- **Additional Tools:** Pandas, NumPy, NLTK/spaCy (as needed)

Code Quality Standards

- Clear project structure with separated modules
- Configuration management (API keys, model settings in config file or environment variables)
- Basic error handling and input validation
- Commented code where logic is complex
- Requirements.txt with all dependencies

Deliverables

Submit the following within 2–3 days:

1. Code Repository

- Complete project folder or GitHub repository link
- Suggested structure:
  - `ingestion.py` – Document loading and chunking
  - `embeddings.py` – Vector embedding generation
  - `rag_pipeline.py` – Retrieval and generation logic
  - `agent.py` – Tool integration and orchestration
  - `api.py` or `app.py` – Interface layer
  - `config.py` – Configuration settings
  - `requirements.txt` – Dependencies
- Include all dummy documents or clear instructions on generating them

2. README Documentation

Comprehensive README.md including:

Setup Instructions:

- Step-by-step guide to install dependencies
- How to configure API keys and environment
- Commands to run the application

Architecture Overview:

- High-level data flow diagram or bullet points:
  - Documents → Chunking → Embeddings → Vector Store
  - Query → Retrieval → LLM Generation → Response
  - Agent/Tool decision flow
- Technology choices and justification

Assumptions & Limitations:

- What assumptions did you make about the data or use case?
- Known limitations of your current implementation

Future Improvements:

- What would you add given more time?
- How would you scale this for production?

3. Sample Queries & Results Report

Create a separate document (Markdown or PDF) with:

- **8–10 example questions** covering different scenarios:
  - Service information queries
  - FAQ-style questions
  - Questions requiring tool usage
  - Edge cases or challenging queries
- For each query, include:
  - The question asked
  - System's answer
  - Sources cited
  - Tool outputs (if triggered)

- Your analysis: "Why this answer is good/acceptable" or "What could be improved"

Example format:

Query 1: "What AI consulting services do you provide?"

Answer: [System response with context]

Sources: services_overview.txt (AI Consulting section)

Analysis: Answer accurately reflects document content with proper citations. Could be enhanced with specific case study references.

Evaluation Criteria

We will assess your submission based on:

Criteria	What We Look For
RAG Implementation	Proper chunking, embedding, retrieval, and generation pipeline
Agentic Behaviour	Thoughtful tool integration with autonomous decision-making
Code Quality	Clean structure, modularity, error handling, documentation
Solution Design	Understanding of business context and practical applicability
Documentation	Clear README, architecture explanation, honest assessment
Testing & Examples	Comprehensive sample queries demonstrating system capabilities
Innovation	Creative approaches, additional features, or optimizations

---

Submission Guidelines

1. Complete the assignment within 2–3 days from receipt
2. Submit via:
    - GitHub repository link (preferred – ensure it's public or add our reviewer)
    - OR ZIP file containing complete project
3. Email submission to: [Insert HR contact email]
4. Subject line: "AI Transformation Assignment – [Your Name]"
5. Include:
    - Link or attachment to code
    - README.md
    - Sample queries report
    - Any additional notes or questions

Important Notes

- You may use any open-source libraries or frameworks
- If using paid APIs (OpenAI), keep costs minimal – we understand token limits
- Focus on demonstrating **understanding and practical skills** over perfection
- If you face blockers, document your approach and reasoning in README
- Authenticity matters – we value your own thinking over copied solutions

---

Questions?

If you have clarifying questions about the assignment, please reach out to:

Contact: [HR Name/Email]
Response Time: We'll respond within 24 hours

Good Luck!

We're excited to see your approach to building AI solutions. This assignment mirrors real work you'd do as part of our team – bridging technical implementation with business value.

Focus on demonstrating:

- Your technical depth in AI/ML and GenAI
- Your ability to design practical, end-to-end solutions
- Your communication through clean code and clear documentation

We look forward to reviewing your submission!

Techculture.ai – AI Transformation Team