POC Documentation Template: RAG Project

# 1. Project Overview

- POC Title: Retrieval-Augmented Generation for [Your Use Case]  
- Team Members:  
- Start Date:  
- End Date:  
- Stakeholders:  
- Business Goal: (e.g., Improve response accuracy for internal knowledge search)

# 2. Problem Statement

Describe the problem this POC aims to solve. Include current limitations or pain points in the existing process.

# 3. Objectives of the POC

List what you aim to demonstrate or validate. e.g.:  
- Validate if RAG can answer domain-specific queries with ≥80% accuracy.  
- Measure latency and feasibility on internal datasets.

# 4. Success Criteria

|  |  |  |
| --- | --- | --- |
| Metric | Target Value | Notes |
| Accuracy | ≥ 80% | Based on SME review |
| Latency | < 3 seconds | Query to response |
| Retrieval Precision | ≥ 0.7 | Cosine similarity/top-k recall |
| User Feedback | Positive | SMEs rate response as helpful |

# 5. Data Sources

List the documents or systems used (e.g., PDFs, manuals, SharePoint, Jira, etc.). Preprocessing steps (e.g., text extraction, chunking logic).

# 6. Architecture

Short description of your pipeline. Optional: Add an architecture diagram (can be created in draw.io, Lucidchart, etc.).

# 7. Tools & Technologies

|  |  |
| --- | --- |
| Component | Tool/Library Used |
| Embedding | e.g., OpenAI, HuggingFace |
| Vector Store | e.g., FAISS, Pinecone |
| LLM | e.g., GPT-4, LLaMA2 |
| Orchestration | e.g., LangChain, LlamaIndex |
| Frontend | e.g., Streamlit (optional) |

# 8. Results & Observations

Summarize test queries and performance. Include sample outputs if relevant. What worked well? What were the limitations?

# 9. Feedback from Stakeholders

Notes or survey responses from business users or SMEs. Summary of usefulness, trust, and areas to improve.

# 10. Conclusion & Recommendation

Was the POC successful? Go / No-Go for production? What would be needed for full implementation?

## ✅ ****Project User Story – BFSI RAG Chatbot for US-Based Client****

### 🧩 ****Problem Statement****

Users — especially in the Banking, Financial Services, and Insurance (BFSI) sector — often need **accurate, real-time answers** pulled from **multiple trusted websites**, including regulatory bodies (e.g., IRS, SEC), banks, and insurance providers.

**Existing chatbots are limited:**

They can't **retrieve and summarize** content from more than one domain in real-time.

They lack **contextual comparison** across financial products or policies.

They often provide **hallucinated** or outdated answers.

### 🎯 ****Solution Overview****

We built a **website-based RAG (Retrieval-Augmented Generation) chatbot** that:

Retrieves live content from multiple websites (e.g., [irs.gov](https://www.irs.gov" \t "_new), [sec.gov](https://www.sec.gov" \t "_new), [chase.com](https://www.chase.com" \t "_new), [statefarm.com](https://www.statefarm.com" \t "_new)).

Uses vector-based semantic search to extract the most relevant chunks.

Synthesizes answers using an LLM (e.g., GPT-4), **with citations**.

Supports real-time comparison between policies, products, or regulatory clauses.

### 👤 ****User Story****

**As a** financial advisor or consumer  
**I want to** ask a question and get answers that reference multiple financial websites  
**So that** I can make informed decisions without manually browsing each site.

### 🔍 ****Example Use Case****

A user types:  
"Compare Roth IRA vs Traditional IRA for 2025. Show limits and tax benefits."

🔹 The chatbot:

Pulls content from the **IRS Roth IRA and Traditional IRA pages** (2025 updated limits).

Summarizes tax treatment differences (pre-tax vs after-tax).

Cites each IRS source and shows a comparison table.

Includes a disclaimer about consulting a tax advisor.

### 🔧 ****Development Steps Followed****

**Requirement Gathering**

Interviewed stakeholders (bank staff, advisors, compliance teams).

Identified high-impact sources (IRS, SEC, CFPB, major banks/insurers).

**Data Source Integration**

Built custom scrapers/crawlers for top financial and regulatory websites.

Indexed them using chunking + vector embeddings (OpenAI + Pinecone).

**RAG Pipeline Setup**

Integrated with FastAPI backend + GPT-4 for response generation.

Set up Top-K retrieval → LLM → context-aware answer generation with citations.

**UI/UX Development**

Built chatbot UI using React with options for:

Web comparison mode

Source highlighting

Export as PDF

**Testing**

Created test queries for retirement planning, loan comparisons, and insurance policies.

Validated answers using SMEs (financial experts).

**Deployment**

Deployed to AWS with CI/CD, integrated with CRM system (Salesforce).

Enabled analytics dashboard for feedback and retraining.

### 🚀 ****How It Helps in Real-Time****

| **Real-Time Feature** | **Benefit to Users** |
| --- | --- |
| 🔄 Live multi-site retrieval | Up-to-date, grounded financial data |
| 📊 Comparison engine | Side-by-side views from different domains |
| 🧠 Contextual LLM responses | Human-like answers with rich insights |
| 📎 Cited responses | Builds trust and ensures traceability |
| 🧰 Agent assist mode | Enhances support team productivity |