Project Report: Enchanted Emporium Gen-AI Assistant

API Choice: Google Generative AI (Gemini)

We chose **Google's Generative AI (Gemini)** API for both:

- **Embeddings**: models/embedding-001 for semantic chunk representation.
- **Chat Model**: gemini-1.5-flash for fast and coherent LLM-based responses.
- Seamless integration via langchain-google-genai.
- Supports both embeddings and chat models with the same API key.
- Gemini 1.5 models offer strong performance at lower latency (ideal for real-time assistant).

Retrieval-Augmented Generation (RAG) Approach

Pipeline:

1. **PDF Ingestion**: Extracted text using PyPDF2.
2. **Chunking**: Used RecursiveCharacterTextSplitter (chunk size: 1000, overlap: 200) to retain context.
3. **Vectorization**: Generated embeddings from each chunk using GoogleGenerativeAIEmbeddings.
4. **Storage**: Stored embeddings using **FAISS**, an in-memory vector database.
5. **Retrieval + QA**: On user query, the top-3 most relevant chunks are retrieved and passed as context to the Gemini chat model using RetrievalQA.

Frontend Design (Streamlit)

- Built a user-friendly UI using **Streamlit**.
- Key features:
  - **PDF Upload Widget** for knowledge base ingestion
  - **Real-Time Chat Input** to ask about potions
  - **Dynamic Chat History Display** (last 5 queries)
  - **Sidebar Instructions** to guide first-time users

⚠ Challenges Faced

1. **API Key Management**:

   a. Ensuring the Gemini API key is passed .
   b. Alternative: Move to. stream lit/secrets.toml for production.
2. **Vector Index Memory Management**:
   a. FAISS is fast but memory-bound; large PDFs may require indexing optimizations or persistent storage.
3. **PDF Quality**:
   a. Extracting text from scanned or image-based PDFs failed silently (since extract text () returns None).
4. **Latency**:
   a. While Gemini Flash is optimized, vector search + LLM inference still takes 1–3 seconds per query.
5. **Chunking Trade-offs**:
   a. Small chunks increase relevance but risk context loss.
   b. Larger chunks retain context but may dilute retrieval precision.

Conclusion

The assistant successfully demonstrates a practical use of RAG with Gemini API to simulate an intelligent, domain-specific recommendation system. Its modular design and clean UI make it extensible for other domains (e.g., recipe books, tech manuals, FAQs)