```
!pip install -q pymupdf sentence-transformers langchain faiss-cpu google-genera
```

```
import os
os.environ["USE_TF"] = "0"

import fitz  # PyMuPDF
from sentence_transformers import SentenceTransformer
from langchain.text_splitter import RecursiveCharacterTextSplitter
import faiss
import numpy as np
import google.generativeai as genai
import re
import time
import json
from typing import List, Dict, Tuple, Optional
from pathlib import Path
import tiktoken
from google.colab import files

print("All imports successful! 📚")
```

```
All imports successful! 📚
```

```
import getpass

print("🔑 Please enter your Google API Key:")
GOOGLE_API_KEY = getpass.getpass("Google API Key: ")

# Check if PDF already exists
pdf_path = "/content/vdoc.pub_java-the-complete-reference-ninth-edition.pdf"

if not os.path.exists(pdf_path):
    print("📄 Please upload your Java PDF textbook:")
    uploaded = files.upload()
    pdf_filename = list(uploaded.keys())[0]
    pdf_path = f"/content/{pdf_filename}"

print(f"✅ PDF ready at: {pdf_path}")
```

```
🔑 Please enter your Google API Key:
Google API Key: ··········
✅ PDF ready at: /content/vdoc.pub_java-the-complete-reference-ninth-editic
```

```
class EnhancedJavaRAG:
    def __init__(self, pdf_path: str, api_key: str):
        self.pdf_path = pdf_path
        self.api_key = api_key
        self.model = SentenceTransformer("all-MiniLM-L6-v2")
        self.chunks = []
        self.chunk_metadata = []
        self.index = None
        self.embeddings = None

        # Validate PDF path
        if not Path(pdf_path).exists():
            raise FileNotFoundError(f"PDF file not found at: {pdf_path}")

        # Configure Gemini
        genai.configure(api_key=api_key)
        self.gemini_model = genai.GenerativeModel(
            "gemini-1.5-flash-latest",
            generation_config=genai.types.GenerationConfig(
                temperature=0.2,
                max_output_tokens=2000,
                top_p=0.8,
                top_k=40
            )
        )

        # Token counter for Gemini
        self.tokenizer = tiktoken.get_encoding("cl100k_base")

    def extract_text_with_metadata(self) -> str:
        """Extract text from PDF with structure preservation."""
        print("📖 Extracting text from PDF...")
        try:
            with fitz.open(self.pdf_path) as doc:
                all_text = ""
                for page_num in range(len(doc)):
                    page = doc.load_page(page_num)
                    blocks = page.get_text("dict")
                    page_text = ""

                    for block in blocks["blocks"]:
                        if "lines" in block:
                            for line in block["lines"]:
                                for span in line["spans"]:
                                    page_text += span["text"] + " "
                                page_text += "\n"
                            page_text += "\n"

                    all_text += f"\n[PAGE_{page_num + 1}]\n{page_text}\n"
                print(f"✅ Extracted text from {len(doc)} pages")
                return all_text
        except Exception as e:
            raise RuntimeError(f"Failed to open PDF: {e}")

    def preprocess_text(self, text: str) -> str:
        """Preprocess text to clean and preserve Java code structure."""
        print("🔧 Preprocessing text...")
        text = re.sub(r'\n\s*\n\s*\n', '\n\n', text)
        text = re.sub(r'[ \t]+', ' ', text)
        text = re.sub(r'(\w)-\s*\n\s*(\w)', r'\1\2', text)
        text = re.sub(r'(\w)([{}();])', r'\1 \2', text)
        text = re.sub(r'([{}();])(\w)', r'\1 \2', text)
        text = re.sub(r'[\u00a0\u2000-\u200b\u2028\u2029]', ' ', text)
        print("✅ Text preprocessing completed")
        return text.strip()

    def create_chunks_with_metadata(self, text: str) -> Tuple[List[str], List[D
        """Create text chunks with metadata for indexing."""
        print("📝 Creating chunks with metadata...")
        text_splitter = RecursiveCharacterTextSplitter(
            separators=[
                "\n\n\n", "\n\n", "\nclass ", "\ninterface ", "\nenum ",
                "\npublic ", "\nprivate ", "\nprotected ", ".\n", ". ", "\n", '
            ],
            chunk_size=1000,
            chunk_overlap=200,
            length_function=len,
            keep_separator=True
        )

        raw_chunks = text_splitter.split_text(text)
        filtered_chunks = []
        chunk_metadata = []

        print(f"📊 Processing {len(raw_chunks)} raw chunks...")
        for i, chunk in enumerate(raw_chunks):
            page_match = re.search(r'\[PAGE_(\d+)\]', chunk)
            page_num = int(page_match.group(1)) if page_match else 1
            clean_chunk = re.sub(r'\[PAGE_\d+\]', '', chunk).strip()

            if len(clean_chunk) < 80:
                continue

            java_keywords = re.findall(
                r'\b(class|interface|extends|implements|public|private|protecte
                clean_chunk, re.IGNORECASE
            )
            code_patterns = re.findall(
                r'(class\s+\w+|interface\s+\w+|\w+\s*\([^)]*\)\s*\{|public\s+\w
                clean_chunk
            )

            chunk_type = "text"
            if len(code_patterns) > 0:
                chunk_type = "code"
            elif len(java_keywords) > 3:
                chunk_type = "technical"

            quality_score = min(len(clean_chunk) / 500, 1.0)
            quality_score += len(java_keywords) * 0.1
            quality_score += len(code_patterns) * 0.2

            filtered_chunks.append(clean_chunk)
            chunk_metadata.append({
                'chunk_id': len(filtered_chunks) - 1,
                'page_number': page_num,
                'length': len(clean_chunk),
                'chunk_type': chunk_type,
                'java_keywords_count': len(java_keywords),
                'java_keywords': java_keywords,
                'code_patterns': code_patterns,
                'quality_score': quality_score,
                'has_code_example': len(code_patterns) > 0
            })

        print(f"✅ Created {len(filtered_chunks)} high-quality chunks")
        return filtered_chunks, chunk_metadata

    def create_embeddings_and_index(self, save_cache: bool = True):
        """Create embeddings and FAISS index, with caching."""
        cache_dir = Path("/content/cache")
        cache_dir.mkdir(exist_ok=True)
        embeddings_file = cache_dir / "enhanced_embeddings.npy"
        chunks_file = cache_dir / "enhanced_chunks.json"
        index_file = cache_dir / "faiss_index.index"

        # Try to load from cache
        if save_cache and embeddings_file.exists() and chunks_file.exists() anc
            try:
                self.embeddings = np.load(embeddings_file)
                with open(chunks_file, "r", encoding='utf-8') as f:
                    cached_data = json.load(f)
                    self.chunks = cached_data['chunks']
                    self.chunk_metadata = cached_data['metadata']
                self.index = faiss.read_index(str(index_file))
                print(f"📁 Loaded cached data: {len(self.chunks)} chunks")
                return
            except Exception as e:
                print(f"⚠️ Cache load failed: {e}. Recreating from scratch...")

        # Create new embeddings
        raw_text = self.extract_text_with_metadata()
        preprocessed_text = self.preprocess_text(raw_text)
        self.chunks, self.chunk_metadata = self.create_chunks_with_metadata(pre

        print("🧠 Creating embeddings...")
        self.embeddings = self.model.encode(
            self.chunks,
            show_progress_bar=True,
            batch_size=32,
            convert_to_numpy=True
        ).astype("float32")

        print("🔍 Creating FAISS index...")
        self.index = faiss.IndexFlatL2(self.embeddings.shape[1])
        self.index.add(self.embeddings)
        print(f"✅ Index created with {self.index.ntotal} vectors")

        # Save to cache
        if save_cache:
            np.save(embeddings_file, self.embeddings)
            with open(chunks_file, "w", encoding='utf-8') as f:
                json.dump({'chunks': self.chunks, 'metadata': self.chunk_metada
            faiss.write_index(self.index, str(index_file))
            print("💾 Cache and FAISS index saved")

    def smart_retrieval(self, query: str, top_k: int = 8) -> List[Dict]:
        """Retrieve relevant chunks with intelligent ranking."""
        if self.index is None:
            raise ValueError("Index not created. Call create_embeddings_and_inc

        print(f"🔍 Searching for: '{query}'")
        query_vector = self.model.encode([query]).astype("float32")
        distances, indices = self.index.search(query_vector, min(top_k * 2, ler

        results = []
        for i, (distance, idx) in enumerate(zip(distances[0], indices[0])):
            chunk = self.chunks[idx]
            metadata = self.chunk_metadata[idx]
            base_score = 1 / (1 + distance)
            relevance_score = base_score

            query_lower = query.lower()
            java_concepts = ['class', 'interface', 'inheritance', 'polymorphism
            if any(concept in query_lower for concept in java_concepts):
                if metadata['java_keywords_count'] > 0:
                    relevance_score *= (1 + metadata['java_keywords_count'] * 0

            if any(word in query_lower for word in ['example', 'code', 'syntax'
                if metadata['has_code_example']:
                    relevance_score *= 1.3

            relevance_score *= (1 + metadata['quality_score'] * 0.2)
            if metadata['chunk_type'] == 'code' and 'example' in query_lower:
                relevance_score *= 1.2
            elif metadata['chunk_type'] == 'technical':
                relevance_score *= 1.1

            results.append({
                'chunk': chunk,
                'metadata': metadata,
                'distance': distance,
                'relevance_score': relevance_score,
                'rank': i + 1
            })

        results.sort(key=lambda x: x['relevance_score'], reverse=True)
        return results[:top_k]

    def generate_context_aware_answer(self, query: str, context: str) -> str:
        """Generate an answer using Gemini with the provided context."""
        # Check token count
        context_tokens = len(self.tokenizer.encode(context))
        if context_tokens > 100000:  # Approximate Gemini limit
            context = context[:int(len(context) * (100000 / context_tokens))]

        enhanced_prompt = f"""You are an expert Java instructor with access to

📚 TEXTBOOK CONTEXT:
{context}

❓ STUDENT QUESTION: {query}

📋 ANSWER REQUIREMENTS:
1. Base your answer ENTIRELY on the provided context
2. Include relevant code examples if present in the context
3. Use proper markdown formatting for code blocks
4. Provide step-by-step explanations when appropriate
5. Reference specific pages when mentioning concepts
6. If the context lacks sufficient information, explicitly state this
7. Structure your answer with clear headings and bullet points

🎯 Please provide a detailed, educational answer:"""

        try:
            print("🤖 Generating answer with Gemini...")
            response = self.gemini_model.generate_content(
                enhanced_prompt,
                safety_settings={
                    'HATE': 'BLOCK_MEDIUM_AND_ABOVE',
                    'HARASSMENT': 'BLOCK_MEDIUM_AND_ABOVE',
                    'SEXUAL': 'BLOCK_MEDIUM_AND_ABOVE',
                    'DANGEROUS': 'BLOCK_MEDIUM_AND_ABOVE'
                }
            )
            return response.text if response and response.text else self._creat
        except Exception as e:
            print(f"⚠️ API Error: {e}")
            return self._create_fallback_answer(query, context)

    def _create_fallback_answer(self, query: str, context: str) -> str:
        """Create a fallback answer from context when API fails."""
        answer = f"# Answer: {query}\n\n"
        answer += "*Based on retrieved textbook content:*\n\n"

        chunks = context.split('---')
        relevant_info = []
        code_examples = []

        for chunk in chunks:
            if len(chunk.strip()) > 100:
                if re.search(r'(class\s+\w+|public\s+\w+\s*\(|import\s+)', chur
                    code_examples.append(chunk.strip())
                else:
                    relevant_info.append(chunk.strip())

        if relevant_info:
            answer += "## Key Concepts:\n\n"
            for i, info in enumerate(relevant_info[:3]):
                answer += f"**Point {i+1}:** {info[:300]}...\n\n"

        if code_examples:
            answer += "## Code Examples:\n\n"
            for code in code_examples[:2]:
                answer += "```java\n"
                answer += code[:500]
                answer += "\n```\n\n"

        return answer

    def ask_question(self, user_query: str, debug: bool = False) -> str:
        """Process a user query and return a context-aware answer."""
        print(f"\n{'='*60}")
        print(f"🎯 QUESTION: {user_query}")
        print(f"{'='*60}")

        relevant_chunks = self.smart_retrieval(user_query, top_k=10)

        if debug:
            print("\n📋 RETRIEVED CHUNKS ANALYSIS:")
            print("-" * 40)
            for i, result in enumerate(relevant_chunks[:5]):
                metadata = result['metadata']
                chunk_preview = result['chunk'][:200] + "..."
                print(f"\n📄 Chunk {i+1}:")
                print(f"   Page: {metadata['page_number']}")
                print(f"   Type: {metadata['chunk_type']}")
                print(f"   Score: {result['relevance_score']:.3f}")
                print(f"   Java Keywords: {metadata['java_keywords_count']}")
                print(f"   Has Code: {metadata['has_code_example']}")
                print(f"   Preview: {chunk_preview}")

        # Build context
        context_chunks = []
        total_length = 0
        max_context = 7000
        used_pages = set()

        for result in relevant_chunks:
            chunk = result['chunk']
            page = result['metadata']['page_number']
            page_bonus = 0.1 if page not in used_pages else 0
            adjusted_score = result['relevance_score'] + page_bonus

            if total_length + len(chunk) <= max_context:
                context_chunks.append({
                    'text': f"[Page {page} | Score: {adjusted_score:.2f}]\n{chu
                    'score': adjusted_score
                })
                total_length += len(chunk)
                used_pages.add(page)
```

```
            if len(context_chunks) >= 6:
                break

        context_chunks.sort(key=lambda x: x['score'], reverse=True)
        final_context = "\n\n---\n\n".join([chunk['text'] for chunk in context_

        if debug:
            print(f"\n📊 CONTEXT STATISTICS:")
            print(f"   Chunks used: {len(context_chunks)}")
            print(f"   Total length: {total_length} chars")
            print(f"   Pages covered: {sorted(used_pages)}")

        print("\n🤖 Generating comprehensive answer...")
        answer = self.generate_context_aware_answer(user_query, final_context)
        print("✅ Answer generated successfully!")
        return answer

print("✅ EnhancedJavaRAG class defined successfully!")
```

```
✅ EnhancedJavaRAG class defined successfully!
```

```
print("🚀 Initializing Enhanced Java RAG System...")

# Initialize the RAG system
rag_system = EnhancedJavaRAG(pdf_path=pdf_path, api_key=GOOGLE_API_KEY)

# Create embeddings and index
print("📚 Setting up knowledge base... (This may take a few minutes)")
rag_system.create_embeddings_and_index()

print("✅ RAG System initialized successfully!")
```

```
🚀 Initializing Enhanced Java RAG System...
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access
  warnings.warn(
```

```
modules.json:  100%                                  349/349  [00:00<00:00,  31.8kB/
                                                    s]

config_sentence_transformers.json:  100%             116/116  [00:00<00:00,  12.3kB/
                                                    s]

README.md:        10.5k/?  [00:00<00:00,  842kB/s]

sentence_bert_config.json:  100%                     53.0/53.0  [00:00<00:00,  5.74kB/
                                                    s]

config.json:  100%                                   612/612  [00:00<00:00,  43.1kB/
```

```
                                                                   s]
model.safetensors: 100%                                  90.9M/90.9M [00:01<00:00, 80.3MB/
                                                                s]
tokenizer_config.json: 100%                                  350/350 [00:00<00:00, 36.7kB/
                                                                     s]
vocab.txt:      232k/? [00:00<00:00, 15.9MB/s]
tokenizer.json:      466k/? [00:00<00:00, 28.6MB/s]
special_tokens_map.json: 100%                                112/112 [00:00<00:00, 7.79kB/
                                                                     s]
config.json: 100%                                          190/190 [00:00<00:00, 13.6kB/
                                                                   s]
📚 Setting up knowledge base... (This may take a few minutes)
📖 Extracting text from PDF...
✅ Extracted text from 1595 pages
🔧 Preprocessing text...
✅ Text preprocessing completed
📝 Creating chunks with metadata...
📊 Processing 2473 raw chunks...
✅ Created 2459 high-quality chunks
🧠 Creating embeddings...
Batches: 100%                                          77/77 [00:05<00:00, 27.55it/s]
🔍 Creating FAISS index...
✅ Index created with 2459 vectors
💾 Cache and FAISS index saved
✅ RAG System initialized successfully!
```

```python
test_query = "What is inheritance in Java with an example?"
print(f"\n🧪 Testing with: {test_query}")

start_time = time.time()
answer = rag_system.ask_question(test_query, debug=True)
end_time = time.time()

print(f"\n📝 TEST ANSWER (Generated in {end_time-start_time:.2f}s):")
print("="*50)
print(answer)
print("="*50)
```

```
🧪 Testing with: What is inheritance in Java with an example?

============================================================
🎯 QUESTION: What is inheritance in Java with an example?
============================================================
🔍 Searching for: 'What is inheritance in Java with an example?'

📋 RETRIEVED CHUNKS ANALYSIS:
```

```
----------------------------------------

📄 Chunk 1:
   Page: 1
   Type: code
   Score: 3.893
   Java Keywords: 14
   Has Code: True
   Preview: . Specifically, the data defined by
the class are referred to as member variables or instance variables . The c
on that data is referred to as member methods or just methods . ( If ...

📄 Chunk 2:
   Page: 1
   Type: code
   Score: 3.608
   Java Keywords: 14
   Has Code: True
   Preview: This chapter examines two of Java’s most innovative features: p
Packages are containers for classes. They are used to keep the class name s
compartmentalized. For example,...

📄 Chunk 3:
   Page: 1
   Type: code
   Score: 3.257
   Java Keywords: 12
   Has Code: True
   Preview: .
In Java, the basis of encapsulation is the class. Although the class will b
detail later in this book, the following brief discussion will be helpful n
the s...

📄 Chunk 4:
   Page: 271
   Type: code
   Score: 2.847
   Java Keywords: 10
   Has Code: True
   Preview: implementation. That is, using interface , you can specify what
it does it. Interfaces are syntactically similar to classes, but they lack
as a ...

📄 Chunk 5:
   Page: 228
   Type: code
   Score: 2.724
   Java Keywords: 6
   Has Code: True
   Preview: Inheritance Basics

To inherit a class, you simply incorporate the definition of one class into
the extends keyword. To see how, let’s begin with a short example. The foll

📊 CONTEXT STATISTICS:
   Chunks used: 6
   Total length: 5524 chars
   Pages covered: [1, 228, 271, 1215]
```

~~~
🤖 Generating comprehensive answer...
🤖 Generating answer with Gemini...
✅ Answer generated successfully!

📝 TEST ANSWER (Generated in 5.24s):
==================================================
## Inheritance in Java

Based on the provided textbook excerpts, inheritance in Java allows a class

**Mechanism:**

*   A subclass incorporates the definition of its superclass using the `ext

**Example (Page 228):**

The textbook provides a concise example of inheritance:

```java
// Superclass A
class A {
    int i, j;
    void showij() {
        // ...
    }
}

// Subclass B inheriting from A
class B extends A {
    void sum() {
        // ... can access i and j directly ...
    }
}
~~~

In this example:

*   `class A` is the superclass.
*   `class B extends A` declares `B` as a subclass inheriting from `A`.
*   `B` inherits `i`, `j`, and `showij()` from `A`.
*   `B` can directly access and use `i` and `j` within its own methods (lik

Important Notes (Page 228):

*   The superclass (`A` in this case) remains an independent, stand-alone c
*   A subclass can also act as a superclass for another subclass, creating

Further Details:

While the provided text explains the basic mechanism of inheritance using t

==
```
```

```
def ask_java_question(question: str, debug: bool = False):
    """
    Simple function to ask a single Java question.

    Args:
        question (str): The Java question to ask
        debug (bool): Whether to show debug information

    Returns:
        str: The generated answer
    """
    try:
        print(f"🔍 Processing question: {question}")
        answer = rag_system.ask_question(question, debug=debug)
        return answer
    except Exception as e:
        return f"❌ Error processing question: {e}"

def interactive_chat():
    """Interactive chat interface for continuous Q&A."""
    print("\n" + "="*60)
    print("🚀 Enhanced Java RAG System Ready!")
    print("="*60)
    print("Ask any Java questions. Type 'quit' to exit.")
    print("-"*60)

    while True:
        try:
            query = input("\n💬 Your Question: ").strip()

            if query.lower() in ['quit', 'exit', 'q']:
                print("👋 Thank you for using the Java RAG system!")
                break
            elif not query:
                continue

            start_time = time.time()
            answer = rag_system.ask_question(query, debug=False)
            end_time = time.time()

            print(f"\n📝 ANSWER (Generated in {end_time-start_time:.2f}s):
            print("-" * 50)
            print(answer)
            print("-" * 50)

        except KeyboardInterrupt:
            print("\n👋 Goodbye!")
            break
        except Exception as e:
            print(f"❌ Error: {e}")

print("\n✅ All blocks executed successfully!")
print("🎯 You can now use:")
print("   - ask_java_question('your question here') for single questions")
print("   - interactive_chat() for continuous chat")
```

```
✅ All blocks executed successfully!
🎯 You can now use:
   - ask_java_question('your question here') for single questions
   - interactive_chat() for continuous chat
```