GenAI Level 2 – Document Search Bot

(AI-Powered Retrieval with Secure Role-Based Access)

Design & Approach Document

Developed by

Ramaksha Kulkarni(27587)

Design & Approach Document

1. Introduction

This document presents a complete design and approach for our Proof of Concept (POC) system, which enables authenticated users to upload documents and ask questions based on their contents. The POC demonstrates end-to-end integration of file handling, security, text extraction, AI prompt construction, and response delivery via a Large Language Model (LLM) API.

2. Purpose

The primary objectives of this POC are:

- **Validate Core Workflows:** Show that users with different roles can securely upload files and retrieve AI-generated answers.
- **Test Text Extraction:** Ensure accurate reading of PDF, DOCX, and TXT files.
- **Demonstrate AI Integration:** Construct effective prompts from extracted text and user queries, handle responses, and implement fallback logic for incomplete answers.
- **Assess Module Design:** Verify that individual components (Auth, File, AI services) interact cleanly and can be extended later.
- Store and retrieve user information securely from a MySQL database

3. Scope

In-Scope

- User registration and login with role assignment (Admin, Power User, Standard User).
- File upload of single PDF, DOCX, or PPTX, XLSX per request.
- Text extraction using Apache Tika.
- Construction of AI prompts from extracted text and user questions.
- Integration with an LLM API (Gemini 1.5-flash) for answer generation.
- Role-based restrictions on viewing and managing Documents and queries.

Out-of-Scope

- Batch or bulk file uploads.
- Real-time collaboration or chat history storage.
- Mobile clients and offline support.

- High-availability or horizontal scaling setups.
- Advanced UI features beyond basic forms and result display.

4. Architecture

The system is divided into discrete modules that communicate over REST APIs within a Spring Boot application. Below is a breakdown of each module, its responsibilities, and internal logic.

4.1 Frontend Module

- **Components:** Login page, Signup page, User Management, Document management, ChatBox(for asking questions).
- **Logic:**
    1. User logs in; frontend stores JWT token via POST /public/login.
    2. Not Registered Signup by default assigned as user via POST /public/Signup
    3. If he is user then can chat only.
    4. If he is an admin, he can view (GET /api/document), upload documents (POST /api/document), delete documents (DELETE /api/document/{filename}).
    5. Admin can view (GET /api/user) Change users to admin and admin to user via GET /api/user/{username}.
    6. File upload form sends file metadata and binary via POST /api/Document.
    7. Question form submits via GET /api/ask/{ question }.
    8. Displays AI response or error message.
    9. Logout explicitly which will remove jwt.

4.2 Security Module

- **Technology:** Spring Security, JWT tokens.
- **Roles & Permissions:**
    - **Admin:** Can manage users, view all uploads and delete documents.
    - **Standard User:** Can query only uploads.
- **Internal Flow:**
    - On login/Signup, PublicController validates credentials and issues JWT and Registers user stores data in database.
    - A JwtFilter intercepts incoming requests, validates the token, and sets SecurityContext.

- Authorization via http method chaining .hasRole('ADMIN') on .requestmatchers() enforces role checks.

4.3 Storage Service Module

- **Responsibilities:** Handle storage and text extraction.
- **Internal Logic:**
  1. Receive file via in multipart format under **2 mb** & **only PDF, DOC, XLS, PPTX**.
  2. Store file in src/main/resources/static/uploads/ folder.
  3. Invoke readAllUploadedFileContents:
     - Checks if format is valid or not.
     - Uses **Apache tika** to read files and extract into String format.
     - Returns it.

4.4 AI Service Module

- **Responsibilities:** Build prompts, call LLM API, process responses.
- **Prompt Construction:**
  1. Retrieve extracted text form storage service.
  2. Prepend a system instruction, e.g., "You are a helpful assistant. Answer based solely on the text provided.".
  3. Append the full extracted text.
  4. Add the user's question at the end.
- **API Call:**
  - Send JSON { prompt: combinedPrompt } to the LLM endpoint.
  - Receive { answer, tokensUsed }.

4.5 Data Flow Diagram (Textual)

[User Interface]

↓

Login (POST /public/login) or Signup (POST /public/signup)

↓

[PublicController] → Store/Retrieve from [MySQL] → Issue JWT

↓

[Frontend stores JWT]

Standard User:

→ GET /api/ask/{question} → [AI Service] → [LLM API] → [Response] → Display Answer

Admin:

→ GET /api/document → View all documents
→ POST /api/document → Upload new document → [Storage Service] → Save + Extract text
→ DELETE /api/document/{filename}→ [Storage Service] → Delete document
→ GET /api/user → [User Service] →View user list
→ GET /api/user/{username} → [User Service] → Toggle user role → Update [MySQL]

5. Technologies Used

Component	Technology
Framework	Spring Boot (Java 17)
Security	Spring Security, JWT Authentication
File Parsing	Apache Tika
AI Integration	Gemini API (1.5-flash)
Database	MySQL
Build Tool	Maven
API Testing	Postman, Swagger UI
File Storage	Local (resources/static/uploads)
IDE	IntelliJ IDEA

6. Assumptions and Limitations

- Files must contain machine-readable text (no image-based PDFs)
- Files over 2MB are not accepted
- Only one file processed per request

- No database persistence for AI results
- LLM output may vary depending on input clarity

7. Future Scope

- **Extended Format Support:** Add OCR for images, support spreadsheets and presentations.
- **Enhanced UI/UX:** Real-time progress indicators, rich text display, and file previews.
- **Role Hierarchies:** Add custom roles and permissions matrix.
- **Persisted History:** Store queries and responses in a database for audit and replay.
- **CI/CD Pipeline:** Automate build, test, and deployment with Docker and Kubernetes.
- **Scalability:** Move text store to a vector database (e.g., FAISS) and cache popular results.
- **Monitoring:** Integrate Prometheus/Grafana for metrics and alerts.

8. Conclusion

This POC successfully demonstrates a basic yet powerful integration of document processing, role-based security, and AI-driven interaction. It validates the feasibility of creating an intelligent assistant system that allows users to gain insights from uploaded documents while ensuring secure access and proper authorization. With the foundation now in place, the system can be extended in several directions, including richer interfaces, better storage and retrieval, and cloud deployment for scalable production use cases.