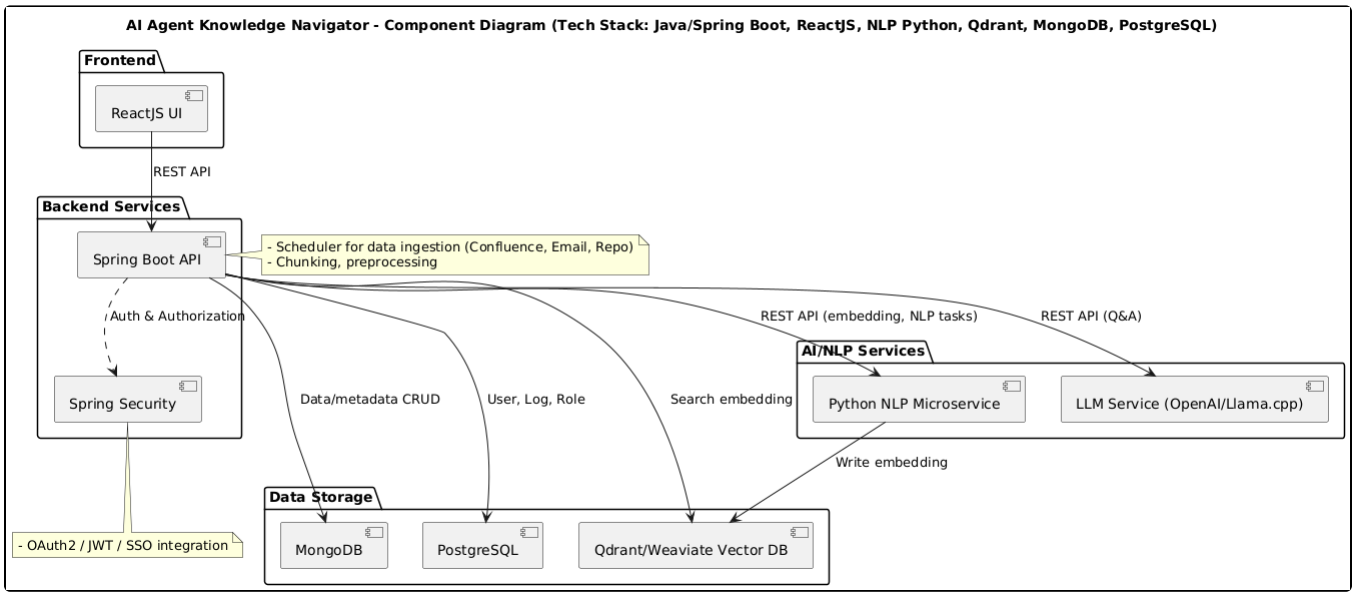
03. Reviewing - Architecture design

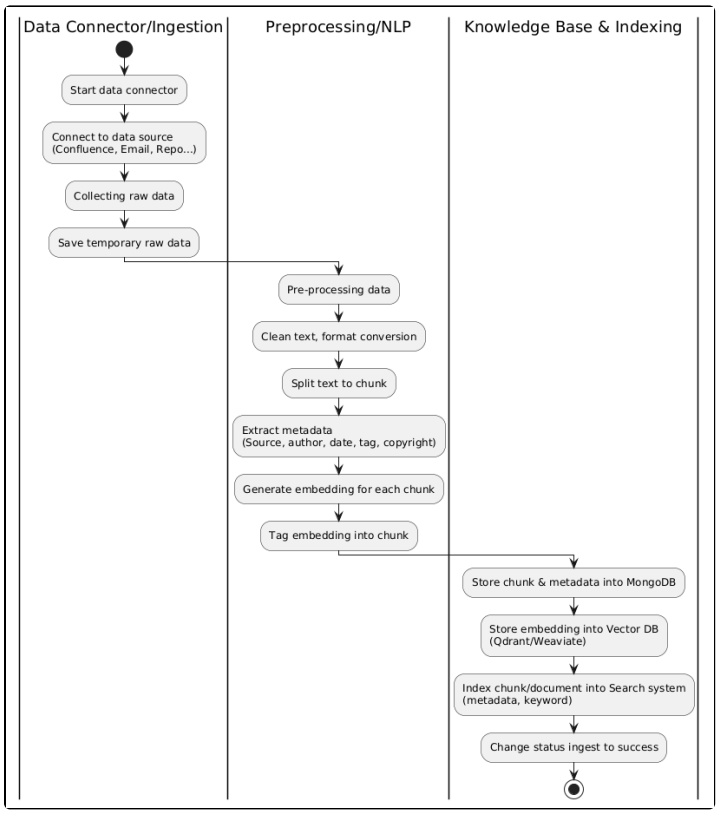
1. Overall architecture



Component	Recommended Technology	Reason for choosing
Data Connector /Ingestion	**Spring Boot, Java**	Easy to integrate API, schedule tasks, easy to maintain with familiar teams.
Preprocessing/NLP	**Java (Stanford CoreNLP, OpenNLP)** hoặc **Python (spaCy)**	Java can be used for basic processing; if advanced NLP is needed, Python microservices can be integrated using spaCy/transformers.
Knowledge Base (Vector Search)	**MongoDB** + **PostgreSQL**	MongoDB stores document chunks & metadata; PostgreSQL stores permissions, logs.
	Qdrant or **Weaviate** (Docker Compose)	Vector DB can run standalone using Docker, exposing API for Java.
Q&A AI Agent /Backend	**Spring Boot, Java**	Build query processing APIs, call out NLP/LLM services.
LLM Q&A Service	**OpenAI API** hoặc **Llama.cpp** (docker, local)	OpenAI API is easy to integrate with Java via REST; Llama.cpp if needed to run locally, connect via API.
Frontend	**ReactJS**	Experienced team, easy to develop modern UI, backend API connection.
Auth & Security	**Spring Security, OAuth2/JWT**	Easy to deploy authentication, user authorization, and integrate SSO if needed.
Deployment & Orchestration	**Docker Compose**	Easily develop, test, package, deploy the entire stack.

2. Processing flow and integration

2.1. Data ingestion



2.1.1. Data Connector/Ingestion

- Building Spring Boot Service to collect data from Confluence, Email, Git repo via API, saved to MongoDB.
- Schedule with Spring Scheduler or Quartz to automatically sync data.

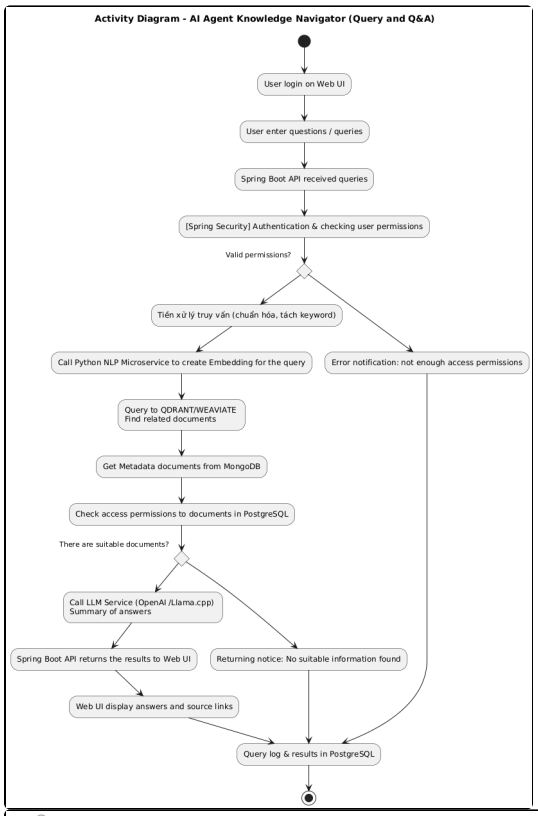
2.1.2. Preprocessing/NLP

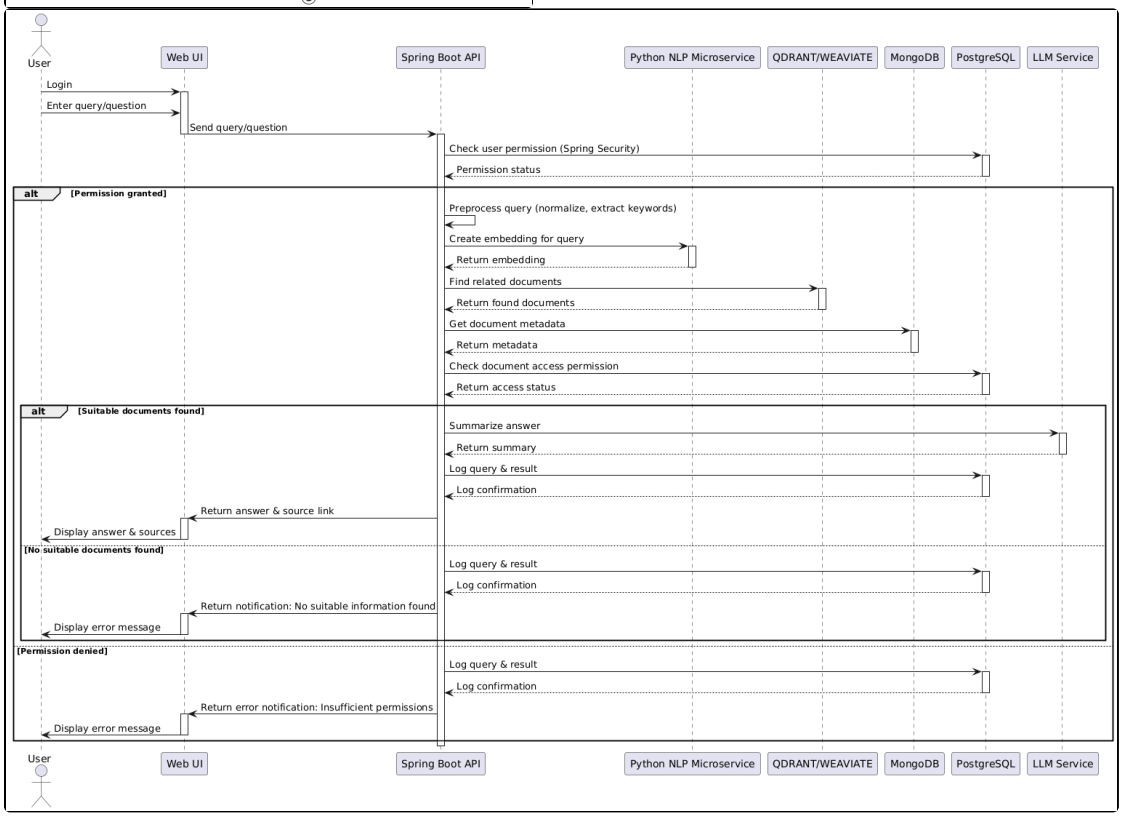
- Use NLP Java libraries (Stanford Corenlp, OpenNLP) to clean, separate, identify keywords.
- If you need Embedding semantic, build a Python Microservice (Spacy, Sentence Transformers) Expose Rest API, Java calls to Embedding.

2.1.3. Knowledge Base & Indexing

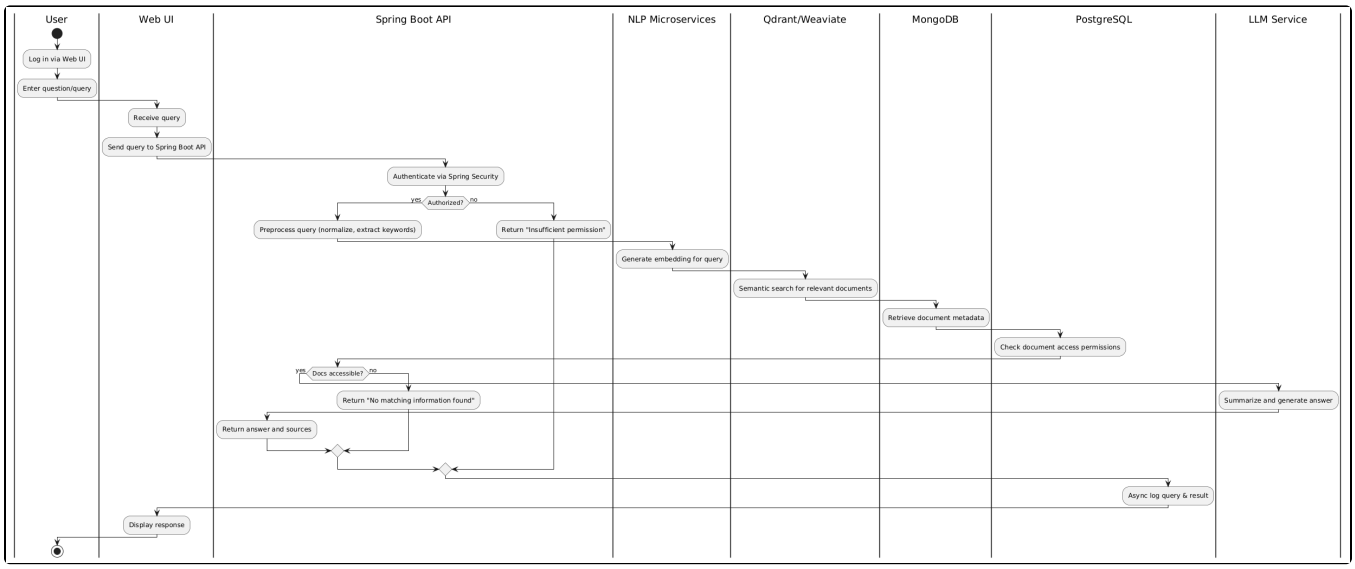
- Save raw data and metadata to MongoDB.
- Save embedding to Qdrant or Weaviate (vector DB), deploy via Docker Compose, expose API for Spring Boot backend to query semantic search.
- PostgreSQL is used to store user information, authorization, and query logs.

2.2. Query and Q&A





Another diagram (swimlane style)



2.2.1. User

- Log in on the web UI
- Enter question/query

2.2.2. Web UI

- Receive query from users
- Send query to Spring Boot API

2.2.3. Spring Boot API

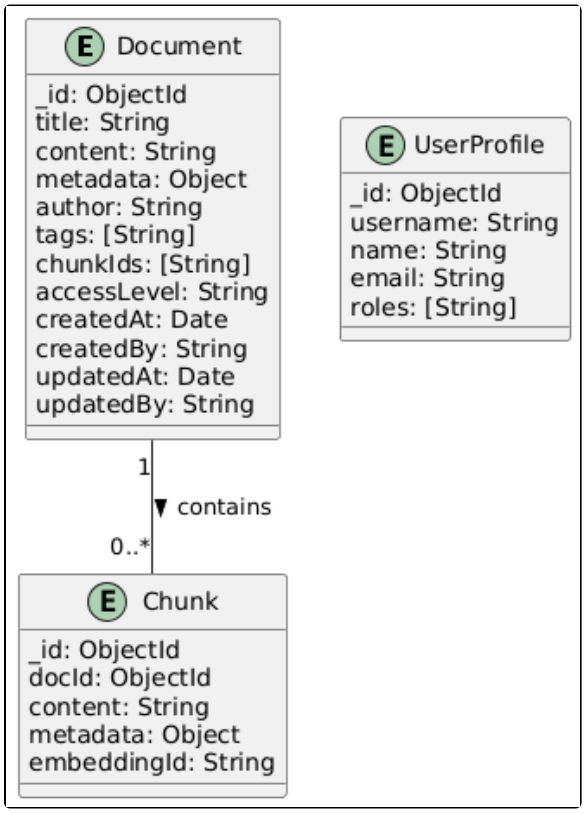
- Receive query
- Authentication via Spring Security (check permissions).
- If enough rights:
    - Query preprocessing (normalization, keyword separation)
    - Call Python NLP Microservice to generate embedding for query
    - Send query to QDRANT/WEAVIATE to find relevant documents
    - Get document metadata from MongoDB
    - Check document access from PostgreSQL
    - If appropriate documentation is available:
        - Call LLM Service (OpenAI/Llama.cpp) to summarize the answer
        - Return results to Web UI
        - Log queries & results to PostgreSQL
    - If no appropriate documentation is available:
        - Return message no matching information found
        - Log queries & results to PostgreSQL
- If not authorized:
    - Returns insufficient permission error message
    - Log queries & results to PostgreSQL

2.2.4. Web UI

- Display results and source links to users

3. Entity

3.1. MongoDB



- **Document**: Store raw document information, metadata, tags, chunk lists.

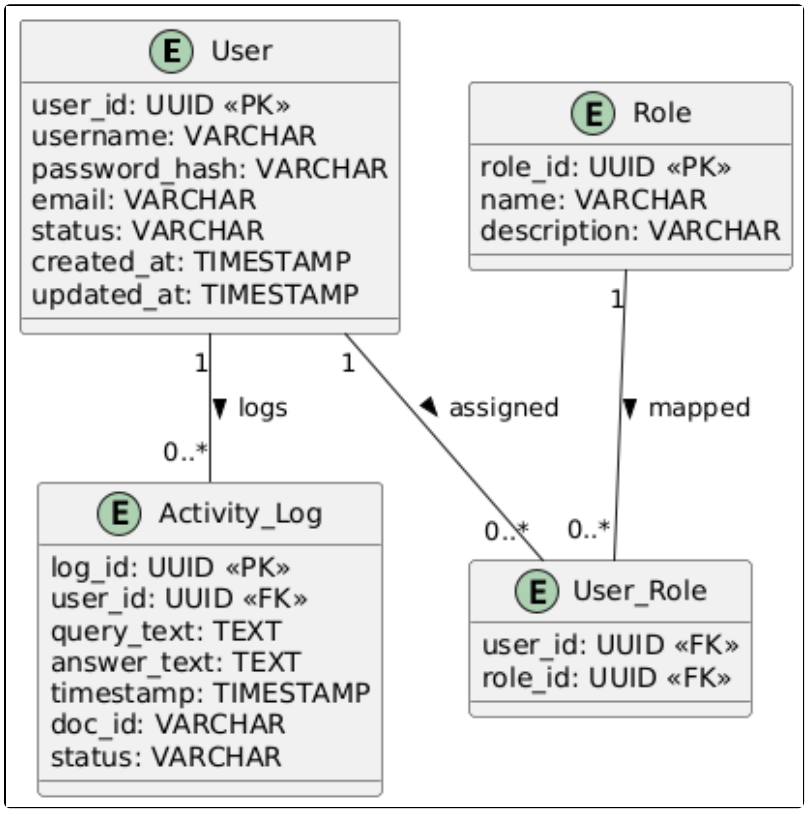
Thuộc tính	Kiểu dữ liệu	Mô tả
_id	ObjectId	Định danh duy nhất cho document
title	String	Tiêu đề tài liệu
content	String	Nội dung gốc của document
metadata	Object	Thông tin phụ trợ (nguồn, ngày tạo, tag…)
tags	Array[String]	Các nhãn phân loại tài liệu
chunkList	Array[Object]	Danh sách các chunk thuộc document
accessLevel	String	Mức phân quyền truy cập tài liệu
createdAt	Date	Thời gian tạo tài liệu
updatedAt	Date	Thời gian cập nhật tài liệu

- **Chunk**: The small document fragment, associated with the document, has an embeddingId.

Thuộc tính	Kiểu dữ liệu	Mô tả
_id	ObjectId	Định danh duy nhất cho chunk
documentId	ObjectId	Liên kết đến document gốc
content	String	Nội dung đoạn nhỏ
metadata	Object	Thông tin phụ trợ về chunk
embeddingId	String	Liên kết embedding trong Vector DB

- **UserProfile**: User information, roles.

3.2. PostgreSQL



- **User**: Login information, email, status.

Thuộc tính	Kiểu dữ liệu	Mô tả
id	UUID	Định danh người dùng
username	VARCHAR	Tên đăng nhập
email	VARCHAR	Email người dùng
password_hash	VARCHAR	Mật khẩu đã mã hóa
status	VARCHAR	Trạng thái tài khoản
created_at	TIMESTAMP	Ngày tạo tài khoản
updated_at	TIMESTAMP	Ngày cập nhật tài khoản

- **Role**: Admin, Developer, Viewer ...

Thuộc tính	Kiểu dữ liệu	Mô tả
id	UUID	Định danh vai trò
name	VARCHAR	Tên vai trò (admin, viewer, developer…)
description	VARCHAR	Mô tả vai trò

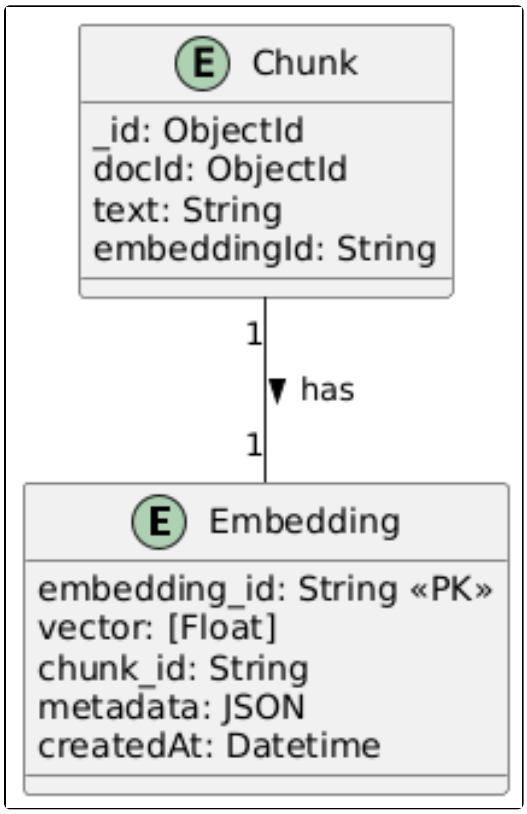
- **User_Role**: Many-to-many relationship between User & Role.

Thuộc tính	Kiểu dữ liệu	Mô tả
user_id	UUID	Liên kết tới bảng User
role_id	UUID	Liên kết tới bảng Role

- **Activity_Log**: Save query history, answer results, documents.

Thuộc tính	Kiểu dữ liệu	Mô tả
id	UUID	Định danh log
user_id	UUID	Người thực hiện truy vấn
query_text	TEXT	Nội dung truy vấn
answer_text	TEXT	Kết quả trả về
document_id	UUID	Liên kết tới document (nếu có)
timestamp	TIMESTAMP	Thời gian thực hiện
status	VARCHAR	Trạng thái (thành công, lỗi…)

3.3. Vector DB (Qdrant/Weaviate)



- **Embedding**: Store vector embeddings for semantic search, associated with chunk/document, metadata.

Thuộc tính	Kiểu dữ liệu	Mô tả
embeddingId	String	Định danh embedding
vector	Array[float]	Dữ liệu vector dùng cho tìm kiếm
chunk_id	String	Liên kết tới chunk/document gốc
metadata	Object	Thông tin phụ trợ (nguồn, vị trí…)
createdAt	DateTime	Thời gian tạo embedding