**Software Requirements Specification**

**for**

**Automatic Document Summarization and Q&A Using LLMs**

**Version 1.0**

**Prepared by**

**Group Name: SYDAI**

|  |  |  |
| --- | --- | --- |
| **Agnishwar Raychaudhuri** | **BT22GDS056** | **agnishwar.raychaudhuri22@st.niituniversity.in** |
| **Gautam Sharma** | **BT22GCS187** | **gautam.sharma22@st.niituniversity.in** |
| **Samarth Tanay** | **BT22GCS192** | **samarth.tanay22@st.niituniversity.in** |
| **Sarabjoth Bhatia** | **BT22GCS233** | **sarabjoth.bhatia22@st.niituniversity.in** |
| **Yuvraj Chawla** | **BT22GCS375** | **yuvraj.chawla22@st.niituniversity.in** |

**Contents**

|  |  |
| --- | --- |
| **Instructor:** | ***Manish Hurkat*** |
| **Course:** | **CS 4191 – Capstone Project II** |
| **Lab Section:** | **D1 D3 D4 D6** |
| **Date:** | **01-02-2025 21/9/25** |

**CONTENTS** [**II**](https://docs.google.com/document/d/1k2atKgNfFJ-okXCJEtOl7c991OcDsLSO/edit#heading=h.49x2ik5)

**REVISIONS II**

**1** **INTRODUCTION** [**1**](https://docs.google.com/document/d/1k2atKgNfFJ-okXCJEtOl7c991OcDsLSO/edit#heading=h.2p2csry)

1.1 Document Purpose [1](https://docs.google.com/document/d/1k2atKgNfFJ-okXCJEtOl7c991OcDsLSO/edit#heading=h.147n2zr)

1.2 Product Scope [1](https://docs.google.com/document/d/1k2atKgNfFJ-okXCJEtOl7c991OcDsLSO/edit#heading=h.3o7alnk)

1.3 Intended Audience and Document Overview [1](https://docs.google.com/document/d/1k2atKgNfFJ-okXCJEtOl7c991OcDsLSO/edit#heading=h.23ckvvd)

1.4 Definitions, Acronyms and Abbreviations [1](https://docs.google.com/document/d/1k2atKgNfFJ-okXCJEtOl7c991OcDsLSO/edit#heading=h.ihv636)

1.5 Document Conventions 2

1.6 References and Acknowledgments [2](https://docs.google.com/document/d/1k2atKgNfFJ-okXCJEtOl7c991OcDsLSO/edit#heading=h.32hioqz)

**2** **OVERALL DESCRIPTION 3**

2.1 Product Overview 3

2.2 Product Functionality [3](https://docs.google.com/document/d/1k2atKgNfFJ-okXCJEtOl7c991OcDsLSO/edit#heading=h.2grqrue)

2.3 Design and Implementation Constraints [3](https://docs.google.com/document/d/1k2atKgNfFJ-okXCJEtOl7c991OcDsLSO/edit#heading=h.vx1227)

2.4 Assumptions and Dependencies 4

**3** **SPECIFIC REQUIREMENTS 5**

3.1 External Interface Requirements 5

3.2 Functional Requirements 5

3.3 Use Case Model 6

3.4 Sequence diagram 14

3.5 Class Diagram 15

**4** **OTHER NON-FUNCTIONAL REQUIREMENTS 16**

4.1 Performance Requirements 16

4.2 Safety and Security Requirements 1[6](https://docs.google.com/document/d/1k2atKgNfFJ-okXCJEtOl7c991OcDsLSO/edit#heading=h.2u6wntf)

4.3 Software Quality Attributes 16

**5** **OTHER REQUIREMENTS 17**

**APPENDIX A – DATA DICTIONARY 18**

**APPENDIX B - GROUP LOG 19**

**Revisions**

| **Version** | **Primary Author(s)** | **Description of Version** | **Date Completed** |
| --- | --- | --- | --- |
| 1.0 | Agnishwar Raychaudhuri  Gautam Sharma  Samarth Tanay  Sarabjoth Bhatia  Yuvraj Chawla | First Draft | 20/09/2025 |

# **Introduction**

## **Document Purpose**

This document specifies the software requirements for "SYD (Summarize Your Documents)," version 1.0. The system is designed to provide users with automated document summarization and an interactive question-answering capability powered by Large Language Models (LLMs). Its purpose is to drastically reduce the time and effort required to extract key information from lengthy documents.

## **Product Scope**

SYD is an intelligent, web-based application designed to address information overload by automating document analysis. The system will allow users to upload documents (PDF, TXT) and, in response, receive high-quality, AI-generated summaries and precise, context-aware answers to natural language questions about the document's content. The core goal is to save users time and enhance their ability to extract critical information efficiently.

## **Intended Audience and Document Overview**

This document is intended for project stakeholders, including the development team (frontend, backend, AI/ML), project managers, and testers. It is structured to provide a comprehensive overview, from high-level project goals to specific technical constraints and functional requirements.

## **Definitions, Acronyms, and Abbreviations**

AI: Artificial Intelligence

API: Application Programming Interface

CI/CD: Continuous Integration / Continuous Deployment

JWT: JSON Web Token

LLM: Large Language Model

OCR: Optical Character Recognition

RAG: Retrieval-Augmented Generation

UI: User Interface

## **Document Conventions**

*This document follows the IEEE formatting standards. The text is written in Arial font, size 11, and is single-spaced with 1-inch margins throughout. Section and subsection titles follow the template provided, with specific conventions for formatting and naming. Italics are used to denote comments or instructions. The document adheres to standard typographical conventions to ensure consistency and readability across all sections. Functional requirements are uniquely identified with the prefix REQ- followed by a number (e.g., REQ-1).*

## **References and Acknowledgments**

* [https://futureagi.com/blogs/revolutionizing-document-management-llm-2025#:~:text=The%20process%20of%20document%20summarization,and%20structure%20of%20the%20document.](https://futureagi.com/blogs/revolutionizing-document-management-llm-2025#:~:text=The%20process%20of%20document%20summarization,and%20structure%20of%20the%20document)
* <https://medium.com/@tahirbalarabe2/summarizing-private-documents-with-llms-langchain-and-rag-3582a67212f2>
* <https://python.langchain.com/docs/tutorials/summarization/>
* <https://docs.paperless-ngx.com>

# **Overall Description**

## **Product Overview**

SYD is a modern, full-stack AI application designed to function as an intelligent research assistant. It leverages a Retrieval-Augmented Generation (RAG) pipeline to provide accurate, context-based answers from user-provided documents. The system is built on a decoupled, containerized architecture to ensure scalability, maintainability, and a robust user experience.

## **Product Functionality**

**User Authentication**: Secure, token-based user login.

**Document Management**: Uploading, processing, and storing documents.

**Text Extraction**: Automated text and data extraction from various document formats, including scanned images (OCR).

**AI Summarization**: Generation of abstractive summaries.

**Interactive Q&A**: A chat-like interface for users to ask questions and receive answers sourced from the document.

## **Design and Implementation Constraints**

The system will be developed and deployed under the following technical constraints:

**Frontend**: The UI must be built as a single-page application using React (with Vite), styled with TailwindCSS.

**Backend**: The primary API must be developed using Python with the Flask framework.

**AI & RAG Pipeline**: The core AI logic must be orchestrated with LangChain, using SentenceTransformers for embeddings, Qdrant as the vector database, and Ollama for local LLM inference.

**Databases**: The system will use PostgreSQL for structured data (users, metadata), MongoDB for unstructured data (chat history), and Redis for caching.

**Infrastructure**: All services must be containerized using Docker and managed locally with Docker Compose. The production environment will use Nginx as a reverse proxy.

**Authentication**: Security must be handled using JWT with refresh tokens, stored in HttpOnly cookies.

## **Assumptions and Dependencies**

The system assumes the availability of open-source models compatible with Ollama (e.g., Llama 3, Mistral).

The functionality is dependent on the performance and accuracy of the chosen embedding and LLM models.

The user is responsible for having the legal rights to upload and process their documents.

# **Specific Requirements**

## **External Interface Requirements**

### **User Interfaces**

A clean, responsive, and intuitive web interface accessible via modern browsers. The UI will consist of a login page, a document upload area, and a main interaction screen with panels for summary display and Q&A chat.

### **Hardware Interfaces**

Users will access the system via desktops, laptops, handhelds like mobile or tablets with internet connectivity (wireless or wired).

The server hosting the web application must support the computational requirements for running the Prediction model.

### **Software Interfaces**

The backend will expose a RESTful API for the frontend. It will also interface with external services like OAuth providers (Google/GitHub) and potentially cloud-based LLM APIs as a fallback.

## **Functional Requirements**

### **REQ-1:** The system shall allow users to authenticate via JWT (email/password or OAuth).

### **REQ-2:** The system shall manage user sessions using JWT with refresh tokens.

### **REQ-3:** The system shall allow users to upload PDF and TXT files.

### **REQ-4:** The system shall use PyMuPDF to extract text from digital PDFs.

### **REQ-5:** The system shall use Tesseract for OCR on scanned documents/images.

### **REQ-6:** The system shall process and clean the extracted text.

### **REQ-7:** The system shall generate vector embeddings for text chunks using SentenceTransformers.

### **REQ-8:** The system shall store and index these embeddings in a Qdrant vector database.

### **REQ-9:** The system shall accept natural language questions from the user.

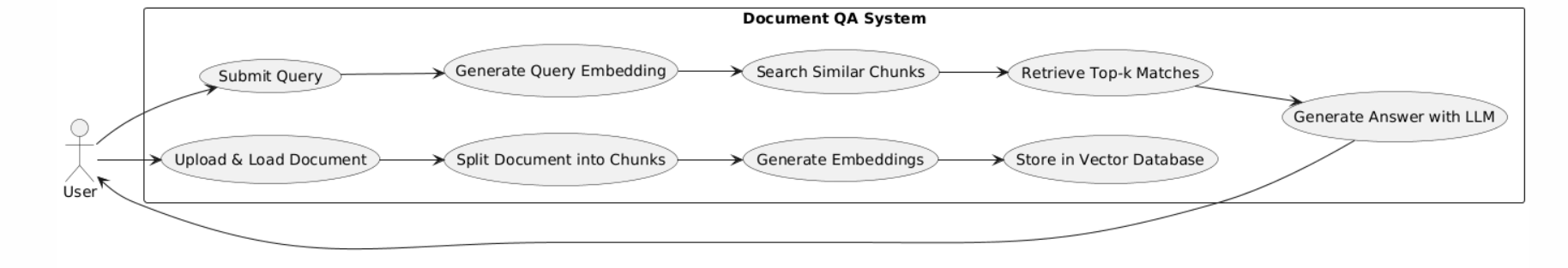
### **REQ-10:** For a given question, the system shall retrieve the most relevant text chunks from Qdrant.

### **REQ-11:** The system shall use LangChain to pass the retrieved chunks and the user's question to an LLM via Ollama.

### **REQ-12:** The system shall display the LLM-generated answer to the user.

### **REQ-13:** The system shall be capable of generating an abstractive summary of the entire document on request.

## **Use Case Model**

****

### **3.3.1 Use Case #1: User Authentication**

**Purpose**:

To allow a user to securely sign in to the system to access their documents and the core features.

**Requirements Traceability**:

REQ-1, REQ-2, REQ-3, REQ-4

**Priority**:  
High

**Preconditions**:

* 1.The user must have a valid account with a supported OAuth provider (e.g., Google, GitHub).
* The SYD system must be online.

**Postconditions**:

* The user is successfully authenticated.
* A secure session (using JWT) is created for the user.
* The user is redirected to their main document dashboard.

**Actors**:

* **Primary Actor:** User

**Flow of Events**:

1. The user navigates to the SYD application homepage.
2. The system presents "Log in with Google" and/or "Log in with GitHub" options.
3. The user selects their preferred provider and is redirected to that provider's login page.
4. After successful authentication with the provider, the user is redirected back to the SYD system.
5. The backend verifies the provider's token, creates a local user profile if one does not exist, and issues a JWT to establish a session. 6. The user is granted access to their personal dashboard.

### **3.3.2 Use Case #2: Upload Document**

**Purpose**:  
To allow an authenticated user to upload a document for analysis, summarization, and Q&A.

**Requirements Traceability**:

REQ-5, REQ-6, REQ-7, REQ-8, REQ-9, REQ-10

**Priority**:  
High

**Preconditions**:

* The user must be authenticated and have an active session.

**Postconditions**:

* The document is securely uploaded to the server.
* The system initiates the background processing pipeline (text extraction, chunking, embedding).
* The document appears in the user's dashboard with a "Processing" status.

**Actors**:

* **Primary Actor**: User

**Flow of Events**:

1. The user clicks the "Upload" button from their dashboard.
2. The system displays a file selection dialog or a drag-and-drop area.
3. The user selects a supported file (e.g., PDF, TXT).
4. The frontend validates the file type and size before sending it.
5. The file is securely transmitted to the backend.
6. The backend acknowledges the upload and adds the document to a processing queue.
7. The user interface updates to show the newly uploaded document and its current status.

### **3.3.3 Use Case #3: Generate Summary**

**Purpose**:  
To allow a user to request and view a concise, AI-generated summary of a fully processed document.

**Requirements Traceability**:

* REQ-11, REQ-12, REQ-13

**Priority**:  
High

**Preconditions**:

* 1. The user must be authenticated. 2. The user must have selected a document that has completed processing (i.e., "Ready" status).

**Postconditions**:

* An abstractive summary of the selected document is generated and displayed to the user.

**Actors**:

* **Primary Actor**: User

**Flow of Events**:

1. The user selects a processed document from their dashboard.
2. The user clicks the "Summarize" button.
3. The user may select a desired summary length (e.g., Short, Medium, Detailed).
4. The backend retrieves the document's content.
5. The content is passed to the LLM with a summarization prompt.
6. The generated summary is returned and displayed in the UI.

### **3.3.4 Use Case #4: Ask a Question**

**Purpose**:  
To allow a user to ask a natural language question about a document and receive a precise, context-aware answer.

**Requirements Traceability**:

REQ-14, REQ-15, REQ-16, REQ-17, REQ-18

**Priority**:  
High

**Preconditions**:

* The user must be authenticated.
* The user must have selected a document that has completed processing.

**Postconditions**:

* A relevant, AI-generated answer based on the document's content is displayed in the chat interface.

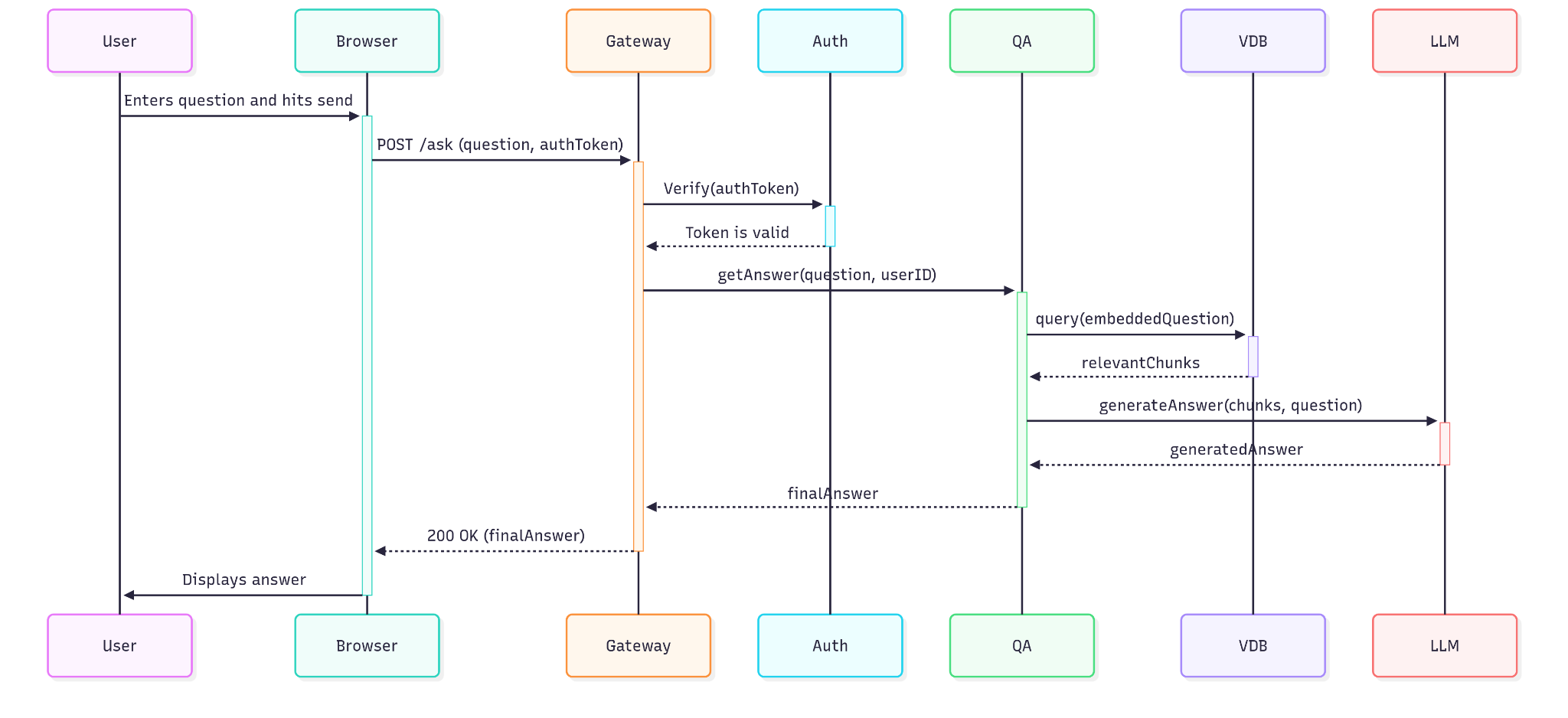
**Actors**:

* **Primary Actor**: User

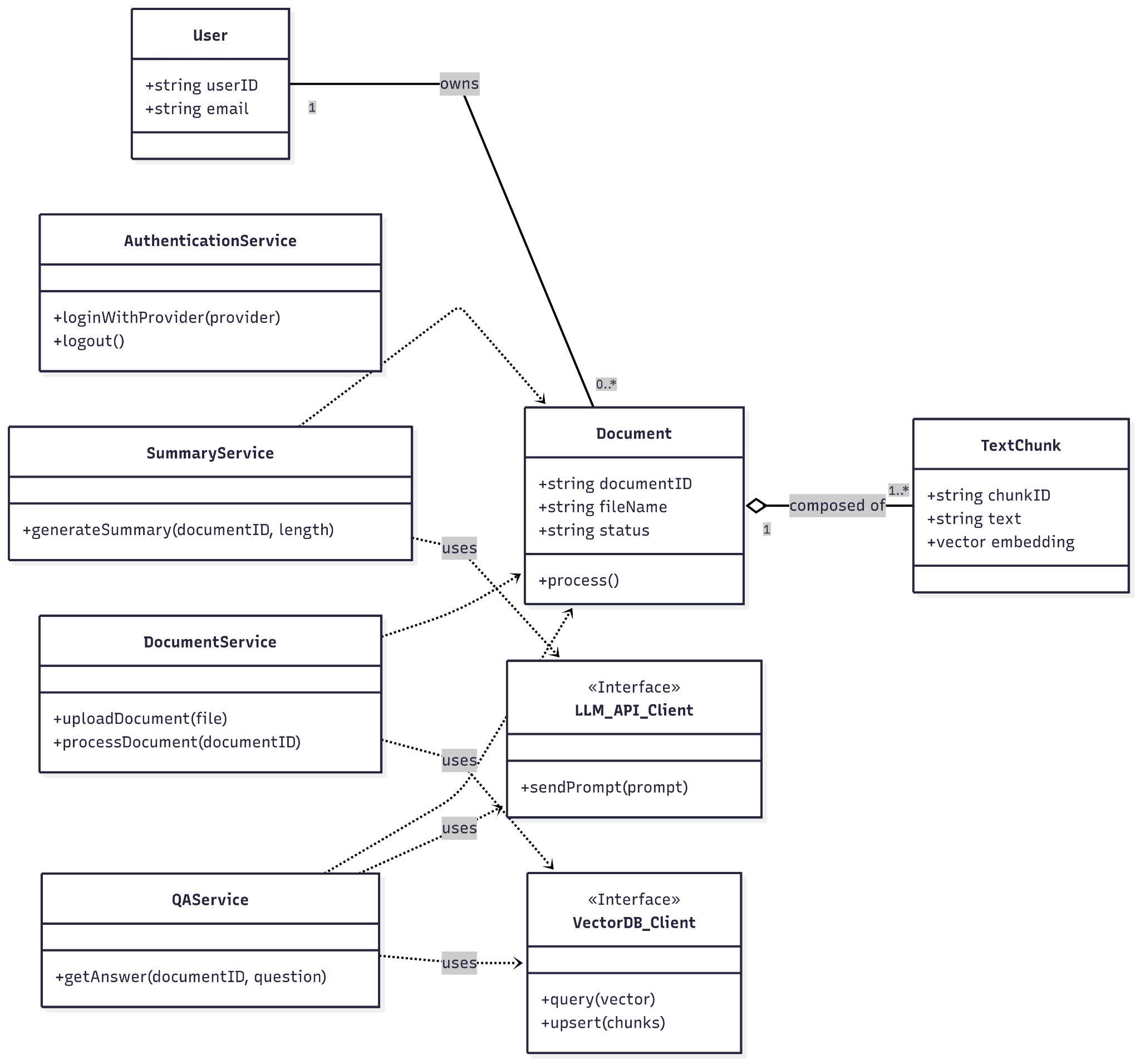
**Flow of Events**:

1. The user selects a document and opens the Q&A interface.
2. The user types a question into the chat input and submits it.
3. The backend generates a vector embedding for the user's question.
4. The system queries the Qdrant vector database to retrieve the most relevant text chunks from the document.
5. The retrieved chunks (context) and the original question are passed to the LLM via the LangChain pipeline.
6. The LLM generates an answer based *only* on the provided context.
7. The answer is sent back and displayed to the user in the chat interface.

**3.4 Sequence Diagram**

****

## **3.5 Class Diagram**



## 

### **3.5.1 Modules**

### **User & Authentication Module**

This module is the entry point to the system and is responsible for all user-related operations. It manages identity, security, and session control by handling user sign-in via OAuth, managing secure sessions with JWT, and validating user permissions for all API requests. The key classes for this module are User and AuthenticationService.

### **Document Management & Processing Module**

This module handles the entire lifecycle of a document from the moment it is uploaded until it is ready for analysis. It manages the secure upload of files (PDF, TXT), extracts raw text (using OCR for scanned images), cleans the text, and breaks it into smaller, semantically meaningful "chunks." The main classes involved are Document, TextChunk, and DocumentService.

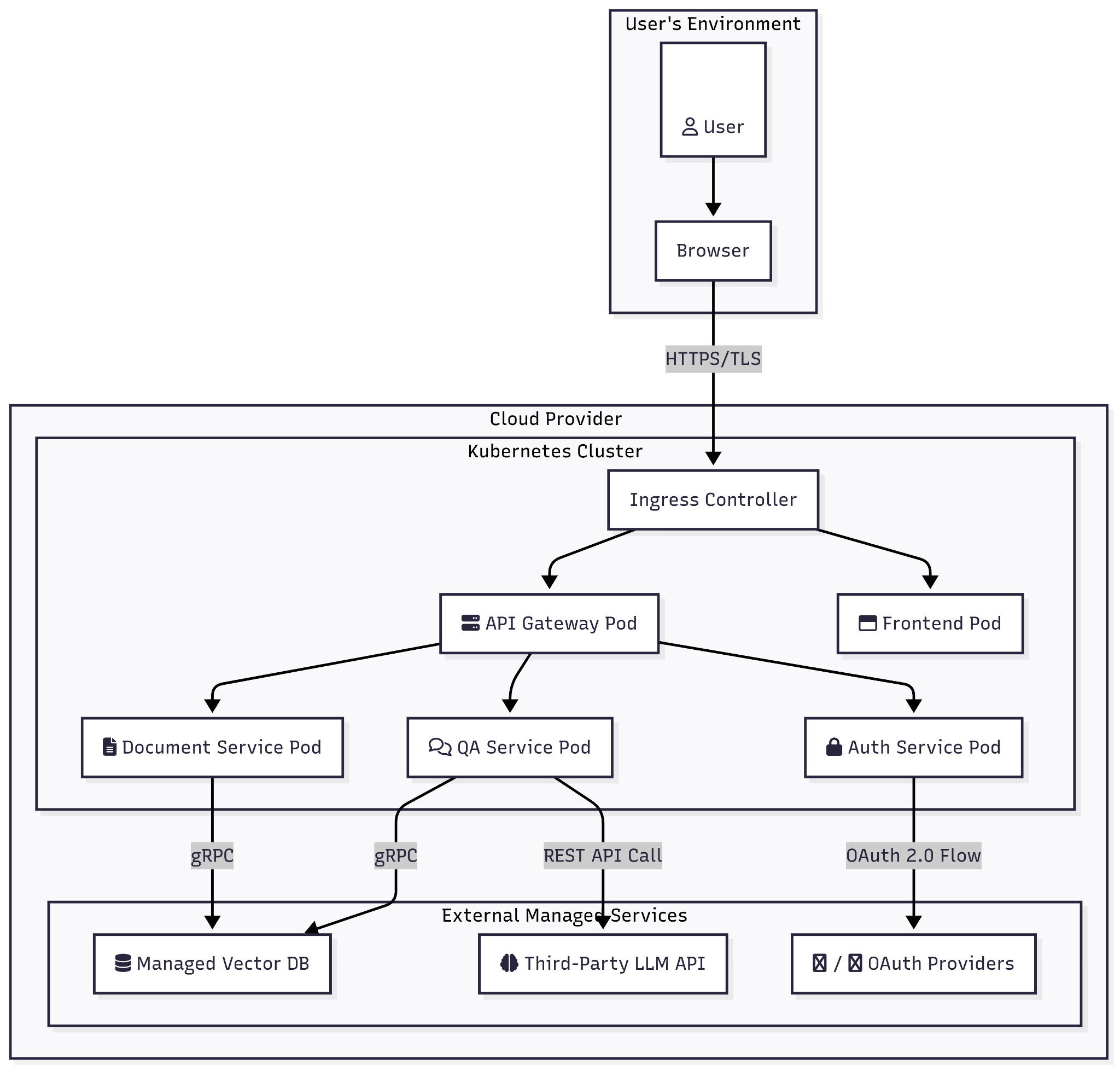
### **AI & RAG Core Module**

This is the intelligent core of the application that performs advanced analysis. It orchestrates the Retrieval-Augmented Generation (RAG) pipeline using LangChain, generates vector embeddings for text chunks, retrieves relevant context from the vector database (Qdrant) in response to a query, and interacts with the Large Language Model (LLM) via Ollama to generate summaries and answers. This module includes the QAService, SummaryService, LLM\_API\_Client, and VectorDB\_Client classes.

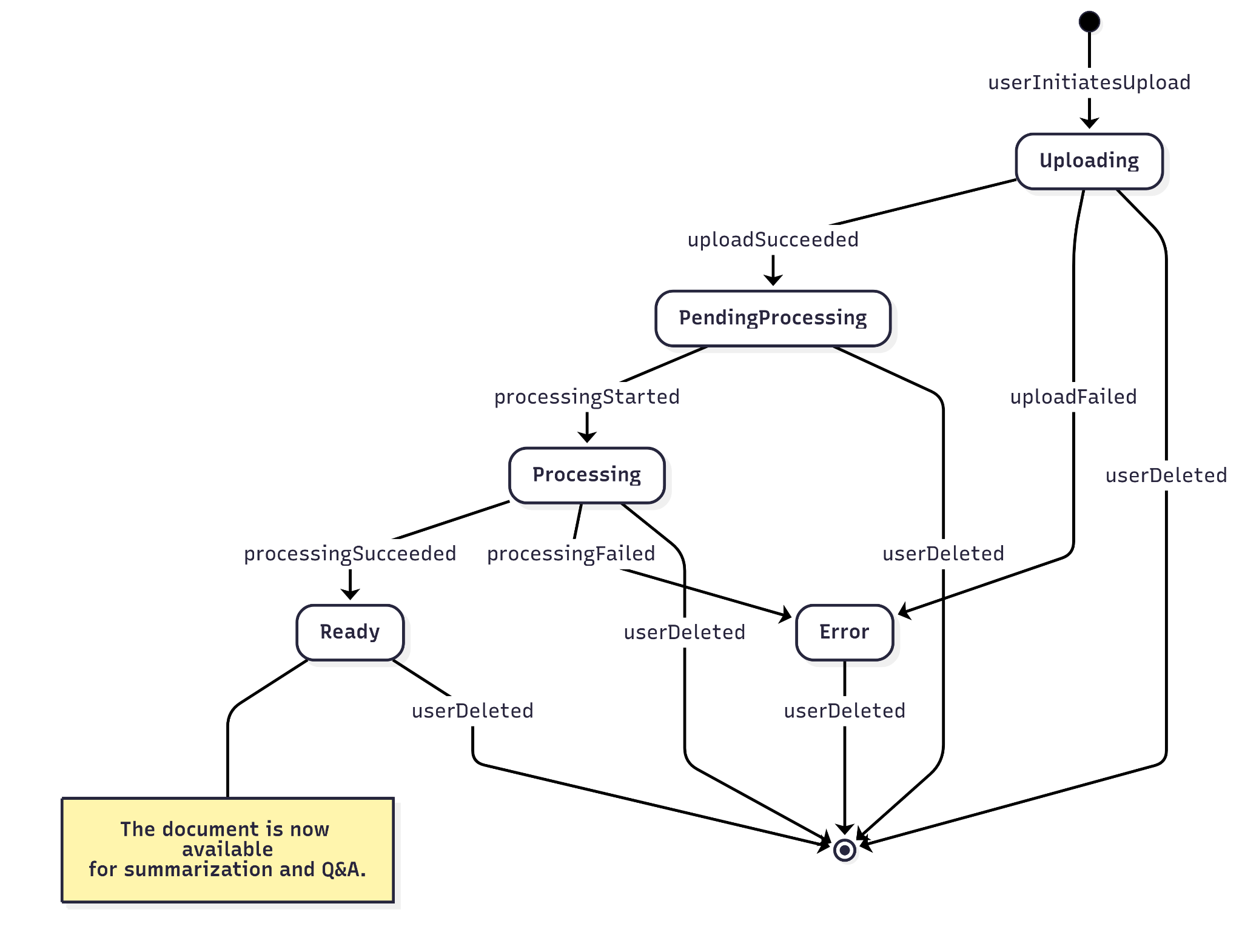
### **Data Persistence & Storage Module**

This module is responsible for all data storage and retrieval, using different databases optimized for specific tasks. It stores structured user and document data in PostgreSQL, manages unstructured chat histories in MongoDB, caches session data in Redis for high performance, and stores all vector embeddings in the Qdrant vector database. This module is primarily composed of the database clients and ORMs (like SQLAlchemy) used by the other services.

## **3.6 Interaction Diagram**



## **3.7 State Diagram**



# **4. Other Non-functional Requirements**

## **4.1 Performance Requirements**

* Q&A and summary generation responses should be delivered to the user in under 15 seconds for a medium-sized document.
* The API endpoints should respond to standard requests within 500ms.

## **4.2 Safety and Security Requirements**

* All web traffic must be encrypted using HTTPS/TLS.
* Passwords must be hashed using bcrypt.
* File uploads must be validated for MIME type and size to prevent malicious uploads.
* The system must implement CORS policies to prevent unauthorized cross-origin requests.

## **4.3 Software Quality Attributes**

**Maintainability:** The codebase will be modular, with a clear separation between the frontend, backend API, and AI pipeline services.

**Reliability:** The containerized architecture ensures that services can be restarted independently, improving overall system uptime.

**Usability:** The user interface will be minimalist and intuitive, requiring no special training to use.

# **5. Other Requirements**

**Appendix A - Glossary**

**Docker:** A platform for building, shipping, and running applications in containers.

**LangChain:** A framework for developing applications powered by language models.

**Ollama:** A tool for running large language models locally.

**Qdrant:** A high-performance vector database used for similarity search.

**RAG (Retrieval-Augmented Generation):** An AI technique that retrieves relevant data from a knowledge base to provide more accurate and context-aware responses.

**Vite:** A modern frontend build tool that provides a faster and leaner development experience for web projects.

**Appendix B - Group Log**