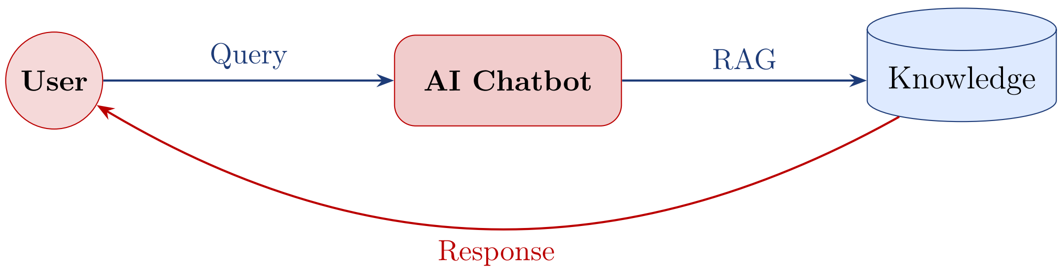
UniVR Chatbot

A RAG-Powered AI Assistant
for the University of Verona



Project Documentation

Version 1.0

Università degli Studi di Verona
January 19, 2026

Contents

1 Project Overview

This short document summarizes the **RAG-based generic chatbot** prototype I implemented. The goal of the prototype is not a finished UniVR product, but a reusable kick-start platform that can support *any* domain by connecting domain documents to a Retrieval-Augmented Generation (RAG) pipeline.

What I Implemented

- A **backend** in FastAPI that exposes chat and admin APIs
- A **frontend** in Next.js that provides a simple chat UI and domain selector
- A **multi-domain RAG layer** built on Google Gemini File Search and File Stores

Instead of hard-coding one university use case, the system lets us create multiple independent domains (e.g. "Scholarships", "Admissions", "International") and query each domain using the same generic chatbot logic.

2 High-Level Architecture

The system follows a simple three-layer architecture:

- **Frontend (Next.js)** — renders the chat UI, lets the user choose a domain, and sends chat requests to the backend.
- **Backend (FastAPI)** — exposes REST endpoints, selects the right domain, orchestrates calls to Gemini and File Search, and returns grounded answers.
- **AI + Knowledge (Gemini + File Stores)** — stores uploaded PDFs per domain, performs retrieval, and powers the RAG responses.

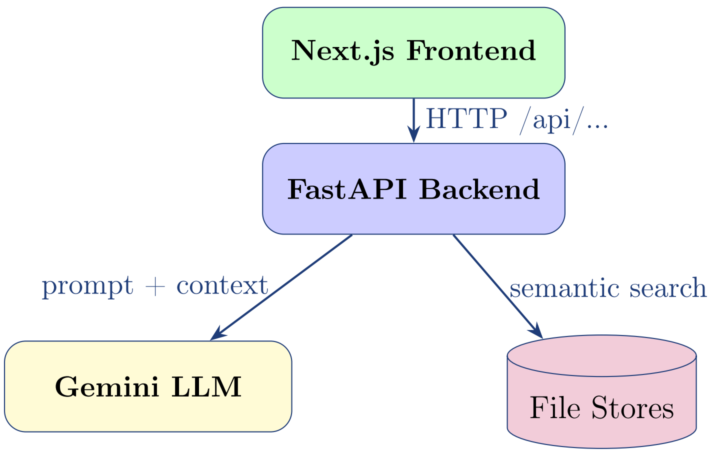


Figure 1: High-level architecture of the generic RAG chatbot

3 RAG Flow Per Domain

Each domain is backed by a separate File Store. The same flow is reused for any domain:

1. **Domain selection**: the frontend calls `GET /api/domains` and shows the available domains returned by the backend.
2. **User question**: the user selects a domain and sends a question via `POST /api/chat` with the domain id.
3. **Retrieval**: the backend uses the configured File Store for that domain to retrieve the most relevant document chunks using Gemini File Search.
4. **Grounded generation**: the backend calls Gemini with the user question plus retrieved passages as context (RAG).
5. **Response**: the backend returns the answer and the sources; the frontend renders them in the chat.

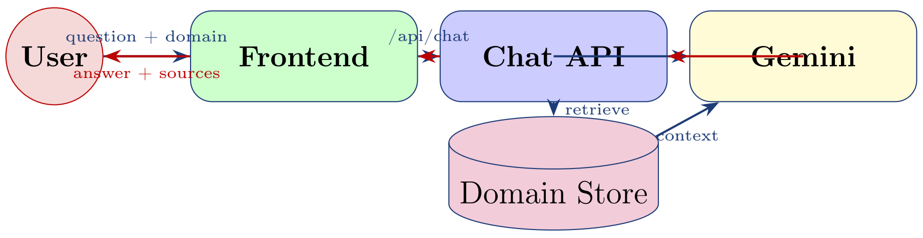


Figure 2: RAG flow for a single selected domain

4 Implementation Summary

4.1 Backend (FastAPI)

- Implemented main application in `app.main` with CORS, static files, and a health check.
- Added **chat API** endpoints under `/api` for: listing domains, sending chat messages, and returning suggestions.
- Implemented an **agent layer** that wraps Gemini: given a domain id and question, it runs retrieval on the matching File Store and builds a grounded prompt for the model.

4.2 Frontend (Next.js)

- Single-page app with: a domain selection view and a chat interface view.
- Uses a configurable `NEXT_PUBLIC_API_URL` so it can talk to any deployed backend (local or on Render).
- Renders the model answer and shows which documents were used as sources.

4.3 Domain Management

- Domains are identified by a prefix and mapped to separate File Stores.
- Admin endpoints allow uploading new PDFs to a domain; the content is indexed by Gemini File Search.
- The same generic chat endpoint works for all domains, only the store id changes.

5 Example Use and Next Steps

5.1 Example: Scholarship Domain

As a concrete example, I created a "Scholarships" domain and uploaded official PDF calls for applications. Students can ask questions such as:

- "What are the main eligibility requirements for this scholarship?"
- "What is the application deadline and where do I submit the form?"

The chatbot answers using the text retrieved from the uploaded PDFs, and the frontend shows which file and section were used.

5.2 Next Steps (After This Prototype)

- Add authentication and simple admin UI for uploading documents and managing domains.
- Improve prompt design and answer formatting for long, legal-style PDF documents.
- Deploy backend (Render) and frontend (Vercel) for stable public access.
- Evaluate quality with real student questions and refine domains and documents.

6 Conclusion

This prototype demonstrates a **generic, multi-domain RAG chatbot** that can be reused for different university domains by simply uploading new documents and creating new stores. The focus of this work is on the integration of FastAPI, Next.js and Gemini File Search into a clean, extensible architecture that can serve as a starting point for a more complete UniVR chatbot in the future.