OmniRAG Project Requirements Specification

This document outlines the key functional and non-functional requirements for the OmniRAG project, serving as a DevOps showcase. These requirements are defined to be clear, unambiguous, and verifiable, guiding the system's design, implementation, and deployment strategies, ensuring the project effectively demonstrates cloud-native DevOps principles.

—

Requirements Summary Table

ID	Requirement Statement	Verification Method
1. Functional Requirements (FR)		
FR-001	The system **shall** accept a natural language query from an external client via a RESTful API.	Submit a query via API and observe system response.
FR-002	The system **shall** retrieve relevant textual information chunks from the pre-processed knowledge base based on the user's query.	For a given query, verify that retrieved chunks are semantically related to the query's intent as assessed by a domain expert.
FR-003	The system **shall** generate a coherent and contextually accurate natural language answer utilizing the retrieved information chunks and the user's query.	Evaluate generated answers against predefined ground truth answers or expert assessment for coherence and factual accuracy (e.g., scoring ≥ 4 out of 5 on a predefined rubric).
FR-004	The system **shall** expose a single, well-documented RESTful API endpoint for all user interactions.	Access API documentation (e.g., Swagger UI) and confirm API endpoint accessibility.
FR-005	The system **shall** process and ingest new or updated documents from a designated input source into the knowledge base, converting them into a searchable vector format.	Ingest a set of new documents and confirm their successful indexing and retrievability via subsequent queries.
FR-006	The knowledge base ingestion process **shall** support '.txt', '.md', and '.pdf' document formats.	Successfully ingest sample documents of each specified format.
2. Non-Functional Requirements (NFR)		
2.1. Performance		
NFR-P-001	The system **shall** generate and return an answer for 90% of user queries within **5 seconds** under typical load conditions (e.g., 5 QPS).	Conduct load testing with simulated user queries and measure response times.
NFR-P-002	The system **shall** sustain a throughput of at least **5 queries per second (QPS)** in the production environment without exceeding NFR-P-001.	Conduct sustained load testing to achieve the specified QPS and confirm latency compliance.
NFR-P-003	The knowledge base ingestion process **shall** process a minimum of **100 documents per hour**.	Ingest a batch of 100 documents and measure the elapsed time.
2.2. Reliability and Availability		
NFR-R-001	The production system **shall** achieve an annual uptime of **99.9%**.	Monitor system availability over a specified period using an external monitoring service.
NFR-R-002	Individual microservice instances (pods) **shall** automatically restart and become operational within **60 seconds** following an unexpected termination event.	Simulate pod termination and observe restart times and health probe status.

ID	Requirement Statement	Verification Method
NFR-R-003	The knowledge base vector embeddings stored in the primary vector database **shall** have a durability of **99.999999999%** (11 nines) over a given year (as per managed service provider SLAs).	Review managed database service's SLA documentation.
2.3. Maintainability and Operability		
NFR-M-001	The system **shall** output all application logs to `stdout/stderr` in a structured format (e.g., JSON), categorized by log level (INFO, WARN, ERROR).	Inspect container logs and confirm format and categorization.
NFR-M-002	Application logs **shall** be aggregated from all running instances and centrally accessible for querying and analysis.	Confirm log streams appear in the centralized logging system (e.g., CloudWatch Logs, ELK stack).
NFR-M-003	The system **shall** expose key application and infrastructure performance metrics (e.g., HTTP request count, latency, error rates, CPU/memory utilization, network I/O, LLM API calls) in a Prometheus-compatible format.	Scrape `/metrics` endpoint and confirm presence of specified metrics.
NFR-M-004	The system **shall** generate automated alerts to designated channels (e.g., Slack, PagerDuty) when critical thresholds are breached (e.g., error rate $> 5\%$, latency > 8 seconds, CPU utilization $> 80\%$).	Simulate error conditions or resource spikes and confirm alert delivery.
NFR-M-005	New application versions **shall** be deployed to development and production Kubernetes environments via an automated CI/CD pipeline without manual steps beyond triggering or approval.	Initiate a code change, observe CI/CD pipeline execution, and confirm successful deployment to the target environment.
NFR-M-006	All cloud infrastructure (e.g., Kubernetes cluster, S3 buckets, ECR repositories) and Kubernetes resources **shall** be defined and managed declaratively as version-controlled code.	Review Git repository for Terraform and Kubernetes manifest files; confirm infrastructure matches code.
NFR-M-007	Application deployments to Kubernetes **shall** follow GitOps principles, where the desired state is defined in Git and reconciled by an in-cluster operator (e.g., ArgoCD).	Initiate a deployment by committing a manifest change to Git and observe ArgoCD's automated synchronization.
NFR-M-008	A repeatable process **shall** exist to set up a local Kubernetes-based development environment, enabling rapid iteration on microservices.	Follow the `environments/local/README.md` instructions to successfully spin up a functional local Kubernetes cluster.
2.4. Security		
NFR-S-001	All sensitive credentials and API keys **shall** be stored and injected into the application securely, not hardcoded in source code or plain-text configuration files.	Inspect application code and configurations; confirm use of Kubernetes Secrets or cloud secret management services.
NFR-S-002	Network access to system components **shall** be restricted to only necessary ports and IP ranges, adhering to the principle of least privilege.	Review network security group rules, Kubernetes Network Policies, and ingress/egress rules.
NFR-S-003	All Docker images built by the CI/CD pipeline **shall** be scanned for known vulnerabilities, and critical vulnerabilities reported.	Review CI/CD pipeline logs for vulnerability scan reports.
2.5. Scalability		

ID	Requirement Statement	Verification Method
NFR-C-001	The system's microservices **shall** be designed to scale horizontally by adding more instances (pods) without requiring code changes.	Successfully scale deployment replicas up and down, confirming stable operation.
NFR-C-002	The production system **shall** automatically adjust the number of running microservice instances (pods) based on CPU utilization and/or custom metrics, to maintain performance under varying load conditions.	Conduct load tests that trigger HPA scaling events, and observe pod count adjustments.