Marmara University

Faculty Of Engineering

CSE3063

OBJECT ORIENTED SOFTWARE DESIGN

TERM PROJECT ITERATION 1

Requirements Analysis Document (RAD)

DATE TO : 21 November 2025 Submitted to: Büşra Arabacı

	Dept	Student Id	Name Surname
1	CSE	150122027	Şeyma Özek
2	CSE	150122060	Fatma Zeynep Kök
3	CSE	150123069	Beyza Çoban
4	CSE	150122007	Beyza Parmak
5	CSE	150123081	Sultan Kocagöz

1. Introduction

1.1 Purpose of the System

This system is designed to provide students with fast, accurate, and officially sourced answers to their questions about academic staff, course/department information, and university regulations. The system processes university documents and automatically generates reliable answers based on the user's query.

1.2 Scope of the System

The system runs entirely via command-line with no GUI, processing all data locally using JSON and YAML files for document chunks, keyword indexes, and configurations, without any database. It uses a deterministic RAG pipeline for intent detection, query rewriting, retrieval, reranking, and cited answer generation. Behavior is configurable via YAML, and all steps are logged in JSONL for transparency. Identical inputs and settings always produce identical outputs.

1.3 Objectives and Success Criteria

The objective of this project is to build a deterministic, modular RAG-style chatbot that answers questions about the Computer Engineering Department, Faculty of Engineering, and university policies using only local text documents. Success is measured by producing identical outputs for identical inputs, correctly detecting intents, retrieving and reranking chunks with deterministic rules, generating valid citations, and completing two CLI scenarios with proper JSONL trace logs. The system must run locally, be fully reproducible via configuration files, and pass the required unit tests.

1.4 References

1. Larman, C. (2004). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development* (3rd ed.). Prentice Hall.
2. Marmara University. (2025). *Computer Engineering Department Official Website*. Retrieved from https://cse-eng.marmara.edu.tr/
3. Marmara University Faculty of Engineering. (2025). *Policies, Directives, and Forms*. Retrieved from https://eng.marmara.edu.tr/idari/basili-belgelerformlar
4. Marmara University. (2025). *University Regulations and Directives*. Retrieved from https://www.marmara.edu.tr/universite/yonetim/mevzuat/

1.5 Overview

This system uses a RAG (Retrieval-Augmented Generation) approach to provide fast and accurate answers to users' university or department-related questions. It processes the query, retrieves and ranks the most relevant document chunks, and generates a meaningful response. Core components—such as the retriever, reranker, intent classifier, and answer generator—work together to deliver accurate information efficiently.

2. Current System

Students search for information manually on the web, which makes the process time-consuming and inefficient.

3. Proposed System

3.1 Nonfunctional Requirements

Usability & Interface: It only works via CLI; configuration and storing are done via YAML/JSON.
Reliability & Performance: The system is deterministic. It generates JSONL logs for debugging. It operates sequentially, without concurrency.
Supportability: Modular structure based on SOLID and GRASP principles. New strategies (NoOpReranker, etc.) can be added via configuration without code changes.
Implementation Constraints: There is no database or GUI. No external APIs are used. It is developed in Java and packaged as a single JAR/Zip.

4. Risks

Missing or incorrect documents: Incorrect, incomplete, or outdated documents may cause the system to produce incorrect answers.

5. Glossary

- **RAG (Retrieval-Augmented Generation):** AI approach generating answers by retrieving document chunks.
- **UC (Use Case):** Scenario describing system-user interaction.
- **JSONL:** Log format with one JSON object per line.
- **CLI / GUI:** Command-Line / Graphical User Interface.
- **Chunk:** Piece of information from source documents.
- **Intent:** Classification of the user's question.
- **Retriever:** Finds relevant document chunks.
- **Reranker:** Reorders retrieved chunks by importance.

UC1	Look up Staff
Primary Actor	Student / System User
Stakeholders	**Student:** Wants to obtain clear and accurate information about academic staff.
	University / Department: Wants official staff information to be provided correctly.
	System: Must correctly classify the question and retrieve the correct chunk.
Preconditions	The system is running and initialized.
	ChunkStore and Config files have been loaded.
	The user provides a meaningful staff-related question.
Postconditions	A trace log entry is written in JSONL format.
	If no valid intent is detected, the system returns “Unknown Intent.”
	The system retrieves accurate staff information and returns it with a valid citation.
Main Success Scenario	The user asks a staff-related question:
	The system calls the IntentDetector.
	IntentDetector determines the intent as **StaffLookup** based on keyword rules.
	The system calls QueryWriter to extract search terms.
	The Retriever browsing ChunkStore and finds relevant chunks.
	The Reranker reorders the hits using proximity and titleBoost rules.
	The AnswerAgent selects the best sentence from the top-ranked chunk.
	The AnswerAgent generates the final answer with citation.
	The system displays the answer to the user.
	All steps are logged to JSONL using the JsonlTraceSink.

Extension	**3a — Intent cannot be determined :**The system returns intent=Unknown → response: "I could not understand your question." **5a — No matching chunks found:** System responds: "No information could be found for the staff member you requested." **6a — All reranker scores tie:** Deterministic tie-break rule applies. **7a — No query terms appear in any sentence:** Fallback: first sentence of the chunk is selected.
Special Requirements	Deterministic output.
	Citation is mandatory.
	JSONL trace logging is mandatory.
	Behavior must be configurable.

UC2	**Look up Course and Department Information**
Primary Actor	Student
Stakeholders	**Student:** Wants correct and up-to-date information about course structure.
	Department: Wants official course data to be served correctly.
Preconditions	The system is running.
	Course-related chunks have already been indexed.
Postconditions	If no intent is detected, return Unknown.
	Trace log is always generated.
	The system returns accurate course/department information with a citation.
Main Success Scenario	The user asks a course-related question:
	IntentDetector classifies the question as **CourseInfo**.
	QueryWriter extracts useful terms.
	Retriever searches ChunkStore for matching chunks.
	Chunks are scored and a Top-K list is created.
	Reranker adjusts the ranking using defined heuristics.
	AnswerAgent selects the best sentence.

	The answer and citation are returned to the user.
	Trace is written to JSONL.
Extension	**2a — Intent mistakenly classified as StaffLookup:** Priority ordering can be configured in YAML. **4a — Question does not contain course code:** QueryWriter falls back to token-based term extraction. **7a — Sentences cannot be split:** Entire chunk may be returned as fallback.
Special Requirements	Deterministic results must be ensured.
	Citation format must follow docId:section:offset.

UC3	**Look up Policy Information**
Primary Actor	Student
Stakeholders	**Student:** Wants a clear explanation of relevant policies or regulations.
	University: Wants official policy text to be relayed accurately.
	System: Must retrieve the correct policy chunk.
Preconditions	Policy documents have been chunked and indexed.
	The system is operational.
Postconditions	Unknown intent may be returned.
	Trace logging always occurs.
	The correct policy statement is returned with a valid citation.
Main Success Scenario	The user asks a policy-related question:
	The system classifies the intent as **PolicyFAQ**.
	QueryWriter extracts relevant terms (e.g., “single”, “course”, “exam”).
	Retriever finds relevant policy chunks.
	Reranker improves ranking using titleBoost and proximity heuristics.
	AnswerAgent selects the best policy sentence.

	Answer + citation are displayed to the user.
	All pipeline stages are recorded in the JSONL trace.
Extension	**2a — Question is vague/general:** Intent=Unknown → “I could not understand what you are asking.” **4a — No relevant policy chunk found:** Response: “No information is available regarding this policy.” **7a — No sentences found:** The full chunk is returned.
Special Requirements	Deterministic pipeline behavior.
	Configurable keyword-based intent rules.
	Citation is mandatory.