Afeka Academic College of Engineering

Department of Computer Science

Detailed Requirements Document

AI-Powered Chatbot for Academic Regulations

Project Team

Omri Roter 206502379

Niv Buskila 209507912

Amitay Manor 20863075

Project Supervisor

Dr. Sharon Yalov-Handzel

Fall Semester
2024

1. Introduction

1.1 Purpose of This Document

This document details the comprehensive requirements for developing an AI-based chatbot system for Afeka College's regulations. It is intended to serve as a complete roadmap for the development team and all stakeholders, ensuring a shared understanding of the system and its requirements.

System Objectives

The proposed system aims to address the ongoing challenge students face in navigating the college's complex academic regulations. By leveraging advanced natural language processing (NLP) techniques and intelligent language models, the system will:

1. Provide immediate, 24/7 answers to students' questions about the college's regulations.
2. Present complex information in a clear and understandable manner.
3. Support queries in Hebrew.
4. Adapt in real-time to policy changes and updates to the regulations.

Scope of the Proposed Solution

The solution includes:

1. Development of an intuitive chat interface based on TypeScript and React.
2. A Python FastAPI-based server system for efficient request processing.
3. Integration of an advanced language model (via NLP/LangChain or equivalent).
4. A flexible storage system based on ChromaDB for storing and retrieving regulatory documents.
5. A comprehensive monitoring and control infrastructure for performance, security, and usability.

Target Audience of this Document

1. Development Team: Developers, UI designers, and ML experts involved in creating the system.
2. Academic Stakeholders: College faculty and administration members who will approve the project and ensure it meets the institution's needs.

This document is written in a detailed manner, allowing each target audience to focus on the relevant sections while maintaining an overall systemic view.

1.2 System Scope

General Description of the System

The chatbot system will enable students to ask questions in natural language and receive precise answers based on the college's regulations. It will improve the user experience and facilitate quick access to relevant information.

System Boundaries

Included:

1. An interactive chat interface for posing questions.
2. A sophisticated RAG pipeline for processing and storing regulatory documents.
3. Vector storage and efficient retrieval mechanisms.
4. Storage and retrieval of information from existing regulations.

Not Included:

1. Editing or modifying the college's regulations.
2. User identity management.
3. Integration with Learning Management Systems.
4. Support for languages other than Hebrew.

Assumptions and Constraints

Assumptions:

1. Digital versions of the regulations are available.
2. Sufficient computing resources are accessible to run language models.

Constraints:

1. The system focuses exclusively on the college's regulations.
2. Answers are limited to information contained in those regulations.
3. An internet connection is required to use the system.
4. The system does not replace individual academic counseling.

1.3 Definitions and Acronyms

Technical Terms

1. API (Application Programming Interface): A set of programming interfaces that allow communication between software components.
2. RAG (Retrieval Augmented Generation): A technique that enhances LLM responses by retrieving and incorporating relevant context from a knowledge base.
3. Embeddings: Vector representations of text that capture semantic meaning.
4. Vector Search: Process of finding similar documents using mathematical distance between vectors.
5. Context Window: Maximum amount of text that can be processed by the language model at once.
6. Chunk Size: The length of text segments created during document processing.
7. Semantic Similarity: Measure of meaning-based relationship between texts.
8. FastAPI: A Python framework for creating efficient APIs.
9. React: A JavaScript library for building dynamic user interfaces.
10. TypeScript: A programming language extending JavaScript with static typing.
11. ChromaDB: A vector database for storing and retrieving textual data semantically.

Business Terms

1. Academic Regulations: Documents defining academic rules and procedures at the college.
2. Query: A user's question or request for information from the system.
3. Response: The system's answer based on the regulations.
4. Accessibility: The process of making complex information clear and accessible to the user.

Acronyms

1. ML (Machine Learning)
2. NLP (Natural Language Processing)
3. UI (User Interface)
4. DB (Database)

2. Overall Description

2.1 System Context

Business Problem:

1. Students have difficulty locating and understanding information in complex regulations.
2. Manual searches through lengthy documents are time-consuming and resource-intensive.
3. The legalistic language of the regulations hinders comprehension.
4. There is a need for quick answers, especially during periods of policy changes.

Current Situation:

1. Regulations are presented as static documents.
2. Information searches are conducted manually.
3. Frequent inquiries are made to academic staff for clarification.
4. Staff are burdened by repeated, similar questions.

Proposed Solution:

1. An intelligent chatbot offering instant responses.
2. Natural language processing of user queries.
3. Precise, semantic search within regulation documents.
4. Clear and simple presentation of information.
5. High adaptability to real-time policy changes.

Integration with the Existing Environment:

1. Accessible through a standard web browser.
2. Utilizes existing digital versions of regulations.
3. Operates independently without reliance on other systems.

2.2 Stakeholders

Stakeholder Identification:

1. Students
    - Primary end-users of the system.
    - Require accurate and quick access to regulatory information.
    - Want to interact using natural language.
2. Academic Staff
    - Seek to reduce workload by minimizing repetitive inquiries.
    - Must ensure the accuracy and reliability of provided information.
3. Development Team
    - Responsible for developing, maintaining, and improving the system.
    - Need access to technical documentation, tools, and stable infrastructure.
    - Must ensure high performance and information security.
4. College Administration
    - Interested in improving student services.
    - Responsible for approving and funding the system.
    - Ensure adherence to academic standards and the college's image.

Needs and Requirements:

1. Students:
    - A user-friendly, accessible interface.
    - Rapid, accurate answers, available 24/7.
2. Development Team:
    - Access to development tools, technical documentation, and regulatory documents.
    - A stable, secure working environment.
    - The ability to monitor, secure, and measure efficiency.
3. Administration and Academic Staff:
    - Reliable and accurate information.
    - Reduced volume of repetitive inquiries.
    - Improved student experience and institutional reputation.

2.3 Key Use Cases

Main Scenarios:

1. Asking a Question About the Regulations:
    - The user enters a Hebrew query in the chat.
    - The system processes the question and identifies its intent.
    - The system returns an answer based on the regulations.
    - The user may ask follow-up questions.
2. Searching for Specific Information:
    - The user searches for information on a particular topic (e.g., enrollment procedures).
    - The system identifies the relevant subject in the regulations.
    - The system displays accurate, relevant information.
    - The user can request expanded information.
3. Clarifying a New Policy:
    - The user asks about a recent policy change (e.g., updated leave-of-absence rules).
    - The system identifies the updated policy.
    - The system shows the changes compared to previous regulations.

Main Process Flows:

1. Asking a Question:
    - Input: A Hebrew question.
    - NLP processing.
    - Searching the regulations database.
    - Composing a relevant answer.
    - Displaying the answer to the user.
2. Updating Regulations:
    - Receiving a new or updated regulation.
    - Processing and storing the information.
    - Updating the search engine and verification.

3. Functional Requirements

3.1 Core Requirements

Mandatory Requirements:

1. Question Processing:
    - Accepting questions in Hebrew.
    - Identifying user intent.
    - Natural language processing (NLP).
2. Information Retrieval:
    - Precise, semantic search within the regulations.
    - Ranking results by relevance.
    - Retrieving information in the proper context.
3. Answer Presentation:
    - Providing a clear, understandable response.
    - Citing relevant regulation sources.
    - Offering additional information upon request.

Key Functions:

1. Conversation Management:
    - Maintaining context throughout the conversation.
    - Allowing follow-up questions.
    - Identifying related topics.
2. Regulation Processing:
    - Ingesting new regulations.
    - Updating existing information.
    - Storing version histories.

Success Criteria:

1. Response Times: Answer within 5 seconds.
2. Accuracy: At least 95% answer accuracy.
3. User Experience: An intuitive interface supporting common browsers.

3.2 Business Processes

Description of Processes:

1. Query Embedding:
    - Convert user question to vector representation.
    - Apply any necessary query preprocessing or expansion.
2. Retrieval:
    - Perform vector similarity search in ChromaDB.
    - Retrieve top-k most relevant chunks.
    - Filter results based on metadata and relevance scores.
3. Context Assembly:
    - Combine retrieved chunks while respecting context window limits.
    - Order chunks by relevance and logical flow.
    - Add necessary system prompts and instructions.
4. Response Generation:
    - Submit assembled context and query to language model.
    - Generate response with citations to source regulations.
    - Validate output against retrieved context.

Business Rules:

1. Information Validity:
    - All answers must be based on approved regulations.
    - The source of the information must be cited.
    - Outdated regulations should not be presented unless the user explicitly requests them.
2. Permissions:
    - Free access for all students.
    - Restricted access to administrative data as needed.

Dependencies and Constraints:

1. Dependencies: Digital regulations, a working language model, and internet connectivity.
2. Constraints: Answers limited to the college's regulations, Hebrew language only, no ability to modify the regulations through the system.

3.3 User Interface (UI/UX)

UI/UX Requirements:

1. General Design:
    - A clean, minimalist interface.
    - Alignment with the Afeka brand identity.
    - Responsive design.
2. Chat Window:
    - A text box for inputting questions.
    - Displaying conversation history.
    - Indicator for question processing.
    - Presenting answers in a readable, clear format.
3. Interaction:
    - A send button.
    - The ability to copy text from answers.
    - Displaying the sources of information.

Main Screens:

1. Main Screen:
    - A main title.
    - Central chat window.
    - Input field for queries.
    - System information.
    - Displaying the current conversation.
    - Presenting excerpts from the regulations.

Navigation Flow:

1. System Entry:
    - Direct access to the main screen and starting a new conversation.
2. Conversation Flow:
    - User inputs a question.
    - System provides an answer.
    - User can continue asking follow-up questions.

3.4 External Interfaces

Interfaces to Other Systems:

1. Regulations Interface:
    - Ingesting regulation files in PDF/Word formats.
    - Automatic update of the system with new regulations.
2. Server Interface:
    - FastAPI-based API.
    - Communication via REST.
    - Secure communication over HTTPS.

Communication Protocols:

1. Client-Server Communication:
    - REST API.
    - JSON data format.
2. Data Storage:
    - ChromaDB for vector storage and retrieval.
    - Storing regulation documents and semantic indexes.

Data Structures in Interfaces:

Example Query Structure:

```
{
 "question": "string",
 "session_id": "string",
 "timestamp": "datetime"
}
```

Example Answer Structure:

```
{
 "answer": "string",
 "source": "string",
 "confidence": "float",
 "timestamp": "datetime"
}
```

4. Non-Functional Requirements

4.1 Performance and Availability

Response Times:

1. Maximum query response time: up to 5 seconds.
2. Main screen load time: up to 2 seconds.
3. Regulation update processing: up to 5 minutes.

Load Handling:

1. Support ~100 concurrent users.
2. Up to ~1000 queries per day.
3. Storing up to 100 regulations in the system.

Required Availability:

1. 99% system availability.
2. Scheduled monthly maintenance window.
3. Recovery time from failure: up to one hour.

Resilience:

1. Daily backups of system data.
2. Log retention for one month.
3. Retention of previous regulation versions for rollback.

4.2 Information Security and Privacy

Security Requirements:

1. Communication Security:
    - Use of HTTPS protocol.
2. System Protection:
    - Firewall and suspicious activity monitoring.
    - DDoS protection measures as needed.

Access Permissions:

1. System Access:
    - Free access for students without complex authentication.
    - Restricted access to the administration interface.
2. System Permissions:
    - Limited access to servers and databases.
    - Separation of development, testing, and production environments.

Data Encryption:

1. Data Encryption:
    - Secure storage of regulations.
    - Encrypted communication between server and client.
2. Key Management:
    - Secure storage of encryption keys.
    - Regular key rotation.
    - Secure backup of keys.

4.3 Usability and Accessibility

Ease of Use:

1. User Interface:
    - Intuitive and simple to operate.
    - Fast response to user actions.
2. User Experience:
    - Clear and understandable error messages.
    - Indication of required fields.
    - Consistent design throughout the system.

Browser Support:

1. Chrome (version 90+).
2. Firefox (version 88+).
3. Edge (version 90+).
4. Safari (version 14+).

4.4 Maintainability and Flexibility

Scalability and Extensibility:

1. Modularity:
    - Modular architecture enabling easy addition of new components.
    - Support for future versions of regulations.
2. Scalability:
    - Ability to handle growing data volume and user numbers.
    - Adding new functionalities without compromising performance.

Ongoing Maintenance:

1. System Maintenance:
    - Regular security updates.
    - Bug fixes and performance improvements.
    - Continuous system monitoring.
2. Updates:
    - Updating new regulations.
    - Updating the language model.
    - Upgrading technology infrastructure.

Required Documentation:

1. Technical Documentation:
    - Code documentation.
    - Architectural documentation.
    - Installation and operation instructions.

5. Architectural and Technological Requirements

5.1 System Architecture

General Architectural Diagram (Textual Description):

1. Client Layer (Frontend):
    - Developed using React with TypeScript.
    - An interactive user interface that communicates with the server via a REST API.
2. Server Layer (Backend):
    - A Python-based FastAPI server.
    - Handles query processing, integrates with the language model, and manages communication with the database.
    - Handles the RAG pipeline including, document processing, embedding knowledge, interacting with chromaDB, retrieving information and generating response.
3. Data Layer:
    - Utilizes ChromaDB for semantic vector storage and retrieval.
    - Stores regulation documents and manages vector-based indexes.

Key Components:

1. Frontend: React components, state management, and REST API client.
2. Backend: FastAPI, NLP modules, and integration with ChromaDB.
3. Database (ChromaDB): Vector embeddings and document storage for semantic search.

5.2 Technological Infrastructure

Development Environment:

1. Tools: VS Code, Git, Docker.
2. Languages: TypeScript (Frontend), Python 3.10+ (Backend), HTML/CSS.

Required Technologies:

1. Frontend: React, TypeScript, Tailwind CSS (or similar UI library).
2. Backend: FastAPI, LangChain (or a similar NLP framework), ChromaDB.
3. Infrastructure: Docker for container management, Git for version control, cloud hosting (Azure, Heroku, etc.).

Operational Infrastructure:

1. Cloud Infrastructure: Application server, database, and file storage.
2. Monitoring and Control: Logging system, performance tracking, and system alerts.

5.3 Data Storage

Data Structure:

1. Regulation Documents: Stored in ChromaDB as text and vectors; version management of regulations is supported.
2. Vectors: Creating embeddings for documents to support semantic search.

Database Type:

1. ChromaDB: A vector database for semantic storage and retrieval of documents.

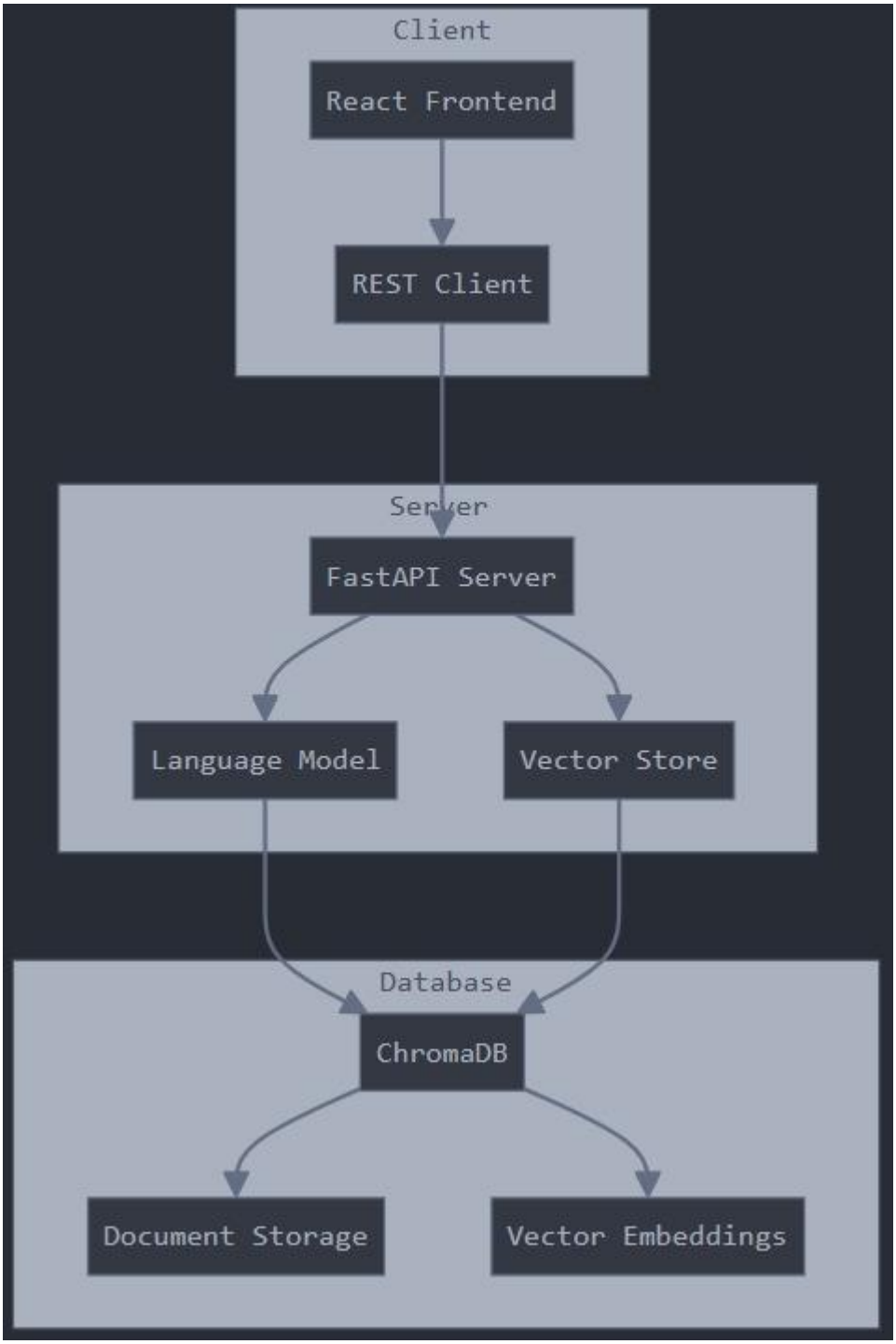
Storage Requirements:

1. Volume:
    - Up to 1GB for regulation documents.
    - Up to 5GB for vector storage.
    - Up to 1GB for logs and operational data.
2. Performance:
    - Retrieval time: up to 2 seconds.
    - Indexing time: up to 1 minute.
    - Regulation update time: up to 5 minutes.

6. Appendices

6.1 Diagrams and Charts

1. Architectural Diagram:



Data Flow Diagram: