Software Requirement Specifications

# AI-Powered Micro SaaS RAG-Based Document Querying System

## Submitted by:

Muhammad Huzaifa (F22BINFT1M01163)

## Submitted to:

Dr. Rana Nazeer Ahmad

Department of Information Technology

Faculty of Computing

The Islamia University of Bahawalpur

Meeting Details

| Sr No | Details | Date | Supervisor Signature |
|---|---|---|---|

Summary

This Software Requirement Specification (SRS) document presents a comprehensive overview of the AI-Powered Micro SaaS RAG-Based Document Querying System. The system integrates NLP, vector embeddings, and a RAG pipeline to allow users to upload documents, process them into embeddings, and query them using natural language. The system retrieves the most contextually relevant chunks and returns an answer generated by a Large Language Model.

Table of Contents

1. Introduction

1.1 Purpose

1.2 Scope

1.3 Product Perspective

1.4 User Characteristics

1.5 Similar Apps and Systems / Literature Review

1.6 Proposed Technologies

2. Requirements

2.1 Functional Requirements

2.1.1 Sign Up

2.2 Non-Functional Requirements

3. Use Cases and Flow of Processes

3.1 Use Case 1

4. References

1. Introduction

Large organizations and academic institutes store data in various documents. Traditional search requires users to read documents manually. This system solves this by allowing users to upload files, process them, and ask natural-language questions. The system uses ChromaDB, embeddings, and an LLM to retrieve and respond with relevant knowledge.

1.1 Purpose

The purpose is to build a scalable Micro SaaS platform where users can upload documents, generate embeddings, and query them using a RAG pipeline.

1.2 Scope

## Includes:

- Uploading PDF, DOCX, TXT, CSV

- Text extraction and chunking

- Embedding generation

- Semantic search

- RAG-based answer generation

## Excludes:

- OCR scanning

- Real-time collaboration

1.3 Product Perspective

## A cloud-hosted microservice integrating:

Frontend (React/Streamlit)

Backend (FastAPI)

ChromaDB

LLM API

1.4 User Characteristics

Admin – manages platform

Registered User – uploads docs and queries

Developer – backend maintenance

1.5 Similar Apps and Systems / Literature Review

ChatPDF – limited customization

Humata AI – proprietary

LangChain QA systems – framework only

1.6 Proposed Technologies

Python, FastAPI, Streamlit/React, SentenceTransformers, ChromaDB, JWT, Docker, PostgreSQL.

2. Requirements

The system processes uploaded documents, extracts text, generates embeddings, stores them, retrieves relevant results, and answers queries using an LLM.

2.1 Functional Requirements

2.1.1 Sign Up

Name: FR001

Purpose: Allows new users to register.

Users: Registered User, Admin

## Input:

- Name

- Email

- Password (8 characters, includes number)

- Phone

- Display Picture (optional)

- Admin Approval

Output: User account created.

## Additional Functional Requirements:

FR002 – Login

FR003 – Document Upload

FR004 – Text Extraction

FR005 – Embedding Generation

FR006 – Embedding Storage

FR007 – Semantic Retrieval

FR008 – RAG Answer Generation

FR009 – Admin Dashboard

2.2 Non-Functional Requirements

- Performance: Query response under 4 seconds

- Scalability: Supports multi-user environment

- Security: JWT, encrypted storage

- Reliability: 99% uptime

- Usability: Clean interface

- Maintainability: Modular code

- Privacy: Document isolation per user

3. Use Cases and Flow of Processes

User uploads doc → system extracts text → chunks text → embeds chunks → stores in ChromaDB → user asks query → system retrieves relevant chunks → LLM generates answer.

3.1 Use Case 1

ID: UC001

Name: Sign Up

Description: Registration process

Requirement: FR001

Actors: User, Admin

Precondition: User not registered

Postcondition: Awaiting approval

## Basic Flow:

1. User enters credentials
2. System validates
3. System creates user

## Alternative Flow:

Admin manually creates user

## Exceptions:

Invalid input, system error

4. References

[1] LangChain Docs

[2] ChromaDB Documentation

[3] SentenceTransformers (Reimers & Gurevych, 2019)

[4] OpenAI API Docs