PRD(Product Requirements Document)는 제품 개발을 위한 요구사항을 정리한 문서로, RAG(Retrieval-Augmented Generation) 시스템을 구축하기 위한 단계별 계획을 체계적으로 작성해보겠습니다. 아래는 간단하면서도 실용적인 PRD 형식으로, FAISS와 Sentence-Transformers를 활용한 저비용 RAG 구현을 목표로 작성했습니다.

# **PRD: RAG 시스템 구축**

## **1. 개요**

### **프로젝트명**

* RAG 기반 지식 검색 시스템

### **목적**

* 외부 데이터를 활용해 정확하고 맥락적인 답변을 제공하는 RAG 시스템을 저비용으로 구축.
* 오픈소스 도구(FAISS, Sentence-Transformers)를 사용해 개발 및 운영 비용 최소화.

### **배경**

* LLM(Large Language Model)의 hallucination 문제를 해결하고, 도메인 특화 지식을 반영한 답변 제공 필요.
* 상용 API 대신 로컬에서 실행 가능한 솔루션으로 초기 투자 및 유지비 절감.

### **목표**

* 개발 기간: 1~2일 (프로토타입 기준).
* 실행 속도: 질문당 1초 이내 응답.
* 데이터: 초기 1000개 문서 기반, 확장 가능성 확보.

## **2. 요구사항**

### **기능 요구사항**

1. **데이터 수집 및 전처리**
   * 입력: 텍스트 파일, PDF 등에서 데이터 추출.
   * 출력: 문서 청크 단위로 분할된 텍스트 리스트.
2. **검색 모듈 (Retrieval)**
   * 입력: 사용자 질문.
   * 출력: 질문과 관련된 상위 1~5개 문서.
3. **생성 모듈 (Generation)**
   * 입력: 검색된 문서 + 질문.
   * 출력: 자연스럽고 맥락적인 답변.
4. **사용자 인터페이스**
   * 최소: CLI (명령줄 인터페이스)로 질문 입력 및 답변 출력.
   * 확장 가능: 웹 UI 고려.

### **비기능 요구사항**

* **성능**: 질문당 응답 시간 1초 이내 (CPU 기준).
* **확장성**: 문서 10만 개까지 지원 가능하도록 설계.
* **비용**: 클라우드 의존 최소화, 로컬 실행 우선.
* **유지보수**: 오픈소스 도구로 모듈 교체 용이.

## **3. 구축 단계별 계획**

### **단계 1: 환경 설정**

* **목표**: 개발 및 실행 환경 준비.
* **세부 작업**:
  1. Python 3.8+ 설치.
  2. 라이브러리 설치:
     + pip install faiss-cpu sentence-transformers transformers.
  3. 하드웨어 확인: CPU 최소 8GB RAM 권장.
* **소요 시간**: 30분~1시간.
* **성과물**: 실행 가능한 Python 환경.

### **단계 2: 데이터 준비**

* **목표**: 검색에 사용할 초기 데이터셋 구축.
* **세부 작업**:
  1. 샘플 데이터 수집 (예: 1000개 문서, PDF 또는 텍스트 파일).
  2. 전처리 스크립트 작성:
     + PyPDF2로 PDF 텍스트 추출.
     + 문서를 500단어 청크로 분할.
     + 불필요한 공백, 특수문자 제거.
  3. 데이터 저장: 리스트 또는 CSV 파일로 정리.
* **소요 시간**: 1~2시간.
* **성과물**: 전처리된 문서 리스트 (예: docs = ["문서1", "문서2", ...]).

### **단계 3: 검색 모듈 구축 (Retrieval)**

* **목표**: 질문에 맞는 문서를 빠르게 검색.
* **세부 작업**:
  1. Sentence-Transformers 모델 로드 (all-MiniLM-L6-v2).
  2. 문서 임베딩 생성 및 저장.
  3. FAISS 인덱스 생성 (IndexFlatL2).
  4. 질문 입력 시 검색 기능 구현:
     + 질문 → 임베딩 → FAISS 검색 → 상위 1개 문서 반환.
* **소요 시간**: 2~3시간.
* **성과물**: 검색 함수 (예: retrieve\_docs(query) → "관련 문서").

### **단계 4: 생성 모듈 구축 (Generation)**

* **목표**: 검색된 문서를 기반으로 답변 생성.
* **세부 작업**:
  1. 경량 LLM 로드 (distilgpt2, Hugging Face Transformers 사용).
  2. 프롬프트 설계:
     + "질문: {query}\n관련 문서: {retrieved\_doc}\n답변:".
  3. 생성 파이프라인 연결:
     + 검색 결과 + 질문 → LLM → 답변 출력.
* **소요 시간**: 1~2시간.
* **성과물**: 생성 함수 (예: generate\_answer(query, retrieved\_doc) → "답변").

### **단계 5: 통합 및 테스트**

* **목표**: 전체 파이프라인 통합 및 동작 확인.
* **세부 작업**:
  1. 검색 + 생성 결합:
     + query → retrieve\_docs → generate\_answer.
  2. CLI 인터페이스 구현:
     + input("질문: ") → print("답변: ").
  3. 테스트 케이스 실행:
     + 질문 5개로 검색 정확도 및 답변 품질 점검.
* **소요 시간**: 1~2시간.
* **성과물**: 동작하는 RAG 프로토타입.

### **단계 6: 최적화 및 확장 (선택)**

* **목표**: 성능 개선 및 대규모 데이터 지원.
* **세부 작업**:
  1. FAISS 인덱스 최적화 (IndexIVFFlat로 전환).
  2. 문서 10만 개 이상 테스트.
  3. 웹 UI 추가 (예: Flask 사용).
* **소요 시간**: 1~2일 (필요 시).
* **성과물**: 확장 가능한 RAG 시스템.

## **4. 일정**

| **단계** | **예상 소요 시간** | **누적 시간** |
| --- | --- | --- |
| 환경 설정 | 30분~1시간 | 1시간 |
| 데이터 준비 | 1~2시간 | 3시간 |
| 검색 모듈 | 2~3시간 | 6시간 |
| 생성 모듈 | 1~2시간 | 8시간 |
| 통합 및 테스트 | 1~2시간 | 10시간 |
| 최적화 (선택) | 1~2일 | 1~2일 (총) |

* **프로토타입 완료**: 약 1일 (8~10시간).
* **확장 포함**: 최대 2~3일.

## **5. 리소스**

### **인력**

* 개발자 1명 (Python, 머신러닝 기본 지식 보유).

### **도구**

* **소프트웨어**: Python, FAISS, Sentence-Transformers, Transformers.
* **하드웨어**: 로컬 PC (CPU 8GB RAM 이상).

### **비용**

* 초기 비용: $0 (오픈소스 및 로컬 실행).
* 확장 시: VPS 또는 GPU 추가 고려 (월 $5~20).

## **6. 성공 기준**

* **기능**: 질문 입력 시 관련 문서 기반 답변 출력.
* **성능**: 초기 구축 15초 이내, 질문당 응답 1초 이내.
* **사용성**: CLI로 최소 5개 질문 테스트 성공.

## **7. 위험 및 대응**

| **위험** | **가능성** | **대응 방안** |
| --- | --- | --- |
| 검색 정확도 낮음 | 중 | Sentence-Transformers 모델 업그레이드 |
| 응답 시간 초과 | 낮 | FAISS 인덱스 최적화 |
| 데이터 전처리 오류 | 중 | 전처리 스크립트 디버깅 강화 |

## **8. 다음 단계**

* 사용자 피드백 수집 후 도메인 특화 데이터 추가.
* 웹 UI 또는 API로 배포.
* 한국어 특화 모델(ko-sbert) 도입 검토.