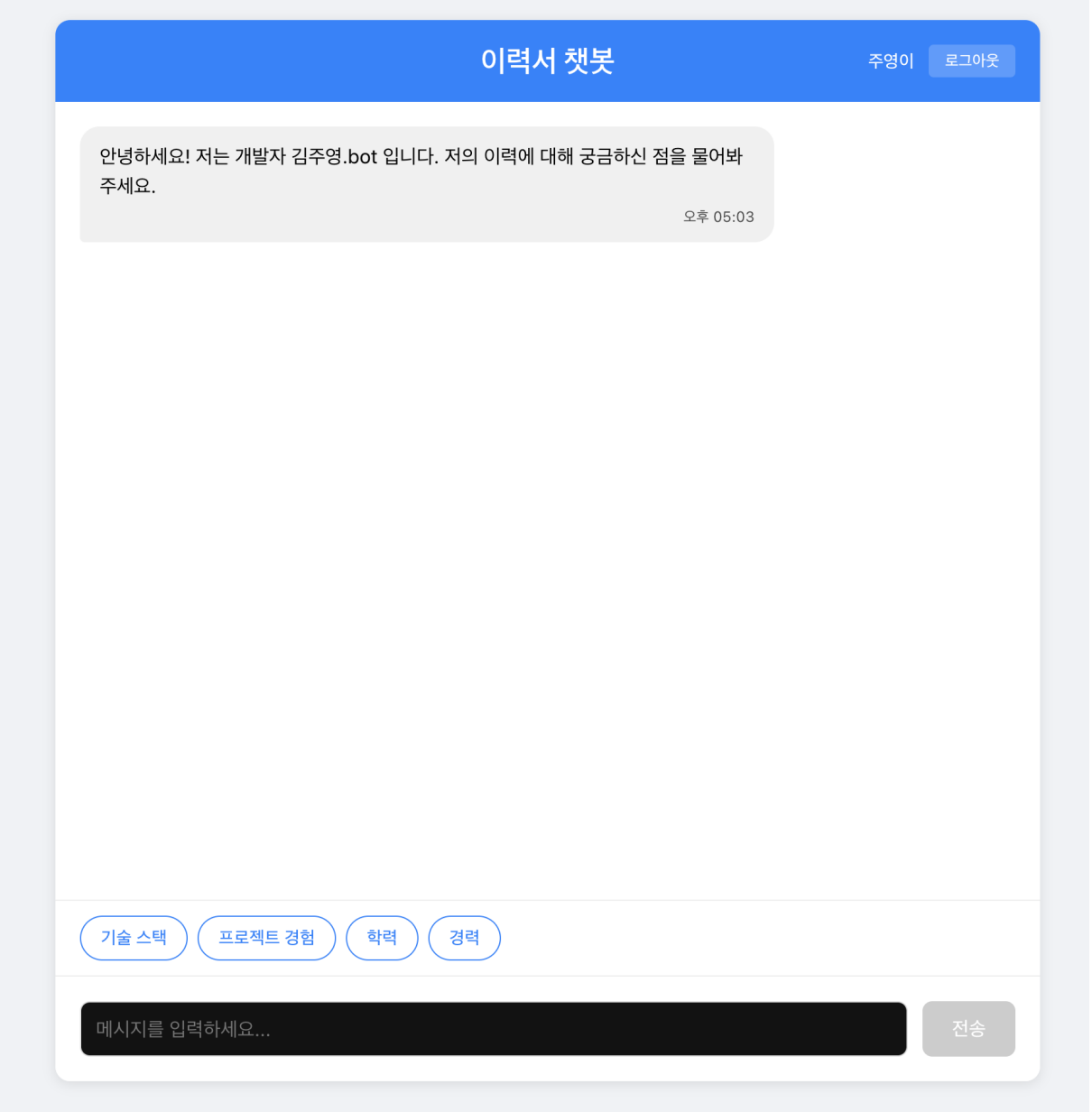
[Personal] Jooyoung.chatbot

프로젝트 주최	본인
규모	1인
날짜 및 기간	2024-12~2025-01
사용 기술	Docker EC2 FastAPI View langchain
설명	본인의 이력서와 포트폴리오를 훈련시킨 개인화 쳇봇



관련 링크

https://github.com/JooyoungOfficielEpitech/portfolio

첫봇 홈페이지

프로젝트 개요

이력서 챗봇은 사용자의 이력서 정보를 자동으로 분석하고, 질문에 대한 실시간 응답을 제공하는 웹 애플리케이션입니다. AI를 활용한 자동화된 질의응답 시스템을 통해 면접관이 지원자와의 직접적인 대화 없이도 인적사항을 알 수 있도록 대화를 지원합니다.

상세 내용

- **설명**:사용자의 이력서를 기반으로 실시간으로 질의응답이 가능한 챗봇 시스템 개발. 프론트엔드와 백엔드를 모두 개발하며, PDF 문서의 자동 분석 및 사용자 경험을 고려한 인터페이스와 API를 설계.
- **핵심 기능**:
  - PDF 이력서 자동 분석 및 저장
  - 사용자 질문에 대한 AI 기반 응답 제공 (Langchain & OpenAI)
  - Redis를 통한 대화 기록 저장 및 관리
  - FastAPI 기반의 백엔드 API 개발 및 최적화
  - 실시간 스트리밍 응답 처리
  - React 기반의 직관적인 사용자 인터페이스 제공

사용 기술 및 라이브러리

- **Frontend**: React, Axios, React Query, Tailwind CSS
- **Backend**: FastAPI, SQLAlchemy, Asyncio, FAISS
- **ETC**:
  - Redis (대화 기록 저장 및 캐싱)
  - Langchain / OpenAI (챗봇 및 문서 분석)

- Docker Compose (배포 환경 구성)
- AWS EC2 (서버 호스팅)
- Github CI/CD (자동 배포 및 테스트)

본인 참여 기능

프론트엔드

- **React 기반 UI/UX 개발**:
  - 사용자 입력 인터페이스 및 실시간 대화 화면 개발.
  - React Query를 활용한 비동기 데이터 최적화.
- **스타일링 및 반응형 디자인**:
  - Tailwind CSS를 사용한 반응형 UI 설계 및 유지보수.
- **API 연동**:
  - Axios를 통한 백엔드 API 연동 및 오류 처리.

백엔드

- **API 개발**:
  - FastAPI를 이용한 RESTful API 설계 및 구현.
  - PDF 분석 및 텍스트 추출 API 개발.
- **데이터 저장 및 최적화**:
  - FAISS를 이용한 유사 이력서 검색 및 질의 응답 최적화.
- **스케줄링 및 비동기 처리**:
  - Redis를 사용한 사용자와의 체팅 기록 기억.
- **챗봇 개발**:
  - Langchain을 활용한 Retrieval-Augmented Generation (RAG) 도입.
  - OpenAI API를 활용한 자연어 처리 응답 최적화.

프로젝트 마무리하며 느낀 점

- **데이터 파이프라인 구축의 중요성**

이력서 데이터를 실시간으로 처리하고 분석하는 과정에서, 데이터의 품질 유지와 적절한 전처리의 중요성을 배웠습니다. 다양한 포맷의 이력서를 표준화하고 일관성을 유지하기 위해 데이터 모델링과 처리 파이프라인을 최적화했습니다.

- **비동기 프로그래밍 및 성능 최적화**

  FastAPI와 Asyncio를 활용하여 다량의 요청을 효율적으로 처리할 수 있도록 설계하였으며, Redis를 통해 대화 히스토리를 캐싱하여 속도를 향상시켰습니다. 이를 통해 API 응답 속도를 단축하고 사용자 경험을 개선할 수 있었습니다.

- **스트리밍 응답의 도입과 구현 어려움**

  사용자가 보다 자연스럽고 끊김 없는 대화를 경험할 수 있도록 `StreamingResponse`를 도입하였으며, 프론트엔드에서 실시간으로 데이터를 처리하고 표시하는 데 있어 상태 관리와 성능 최적화의 중요성을 배웠습니다.

- **자동화된 배포 및 운영 경험**

  Docker 및 AWS EC2를 이용한 컨테이너 기반 배포를 진행하면서, CI/CD 파이프라인을 구축하여 자동화된 배포 환경을 마련하였고, 안정적인 서비스 운영에 필요한 모니터링 및 로깅을 경험했습니다.

- **사용자 중심의 UI/UX 설계**

  사용자 입장에서 직관적인 인터페이스와 원활한 대화 흐름을 설계하는 것이 기능 구현만큼 중요하다는 점을 깨달았으며, 이를 위해 피드백을 반영하고 반복적인 테스트를 수행했습니다.

- **자연어 처리(NLP) 활용의 도전과 한계**

  OpenAI API를 활용하여 질의응답을 제공하면서, 특정 도메인에서의 응답 정확도를 높이기 위해 커스텀 프롬프트 엔지니어링과 데이터 학습이 필요함을 실감했습니다.