

머신러닝, 자연어 처리 기반 중고차 추천 플랫폼



202012142 이은진

201824606 최희웅

201913543 최성빈

지도교수 유영환

목 차

1. 서론	1
1.1. 연구 배경	1
1.2. 연구 목표	2
2. 연구 배경	2
2.1. 배경 지식	2
2.1.1. LLM(Large Language Model)	2
2.1.2. RAG(Retrieval-Augmented Generation)	3
2.2. 기술 스택	6
2.2.1. 크롤링	6
2.2.2. Flask	6
2.2.3. Docker	7
2.2.4. 백엔드	7
2.2.5. 프론트엔드	8
3. 연구 내용	9
3.1. 챗봇 기본 설계	9
3.1.1. 시스템 구조도	9
3.1.2. 챗봇 UI 디자인	10
3.1.3. 데이터베이스 설계	11
3.2. RAG 시스템 구축	12
3.2.1. 중고차 데이터 크롤링	12
3.2.2. 중고차 관련 정보 수집	13
3.2.3. Langchain 기반 구현	14
3.3. 모델 API 구현	21
3.3.1. Flask	21
3.3.2. Docker	22
3.4. 백엔드 개발	22
3.4.1. 패키지 재구성 및 수정	22
3.4.2. API 테스트	23
3.5. 프론트엔드 개발	24
4. 연구 결과 분석 및 평가	28
5. 결론 및 향후 연구 방향	28
6. 구성원별 역할 및 개발 일정	30
6.1. 구성원별 역할	30
6.2. 개발 일정	31
7. 참고 문헌	32

1. 서론

1.1. 연구 배경

오늘날 AI 기술의 발전은 다양한 분야에서 혁신을 가져오고 있으며, 특히 LLM(Large Language Model)과 머신러닝(ML)이 그 핵심 역할을 하고 있다. LLM은 대규모 텍스트 데이터를 학습하여 자연어를 이해하고 생성하는 모델로, 대표적인 예로는 OpenAI의 chat GPT나 Meta의 LLaMA등을 들 수 있다. 머신러닝은 데이터를 분석해 예측 및 결정을 내리는 알고리즘으로, 스팸 필터링, 이미지 인식, 추천 시스템 등 다양한 분야에서 활용된다.

특히 IR(Information Retrieval, 정보 검색) 분야에서는 AI의 발전이 새로운 변화를 가져왔다. 기존 검색 엔진은 키워드 기반의 문서나 정보를 제공하는 방식이었지만, LLM 기반의 대화형 AI 챗봇은 사용자와의 대화를 통해 더 자연스럽게 맞춤형으로 정보를 제공하는 새로운 패러다임을 제시하고 있다.

그러나 이러한 챗봇은 일반적인 지식을 바탕으로 작동하기 때문에 특정 도메인이나 전문 분야의 깊이 있는 정보 제공에는 한계가 있다. 예를 들어, 의료, 법률, 금융과 같은 전문 분야에서는 정확하고 세밀한 정보가 요구되지만, 일반적인 LLM은 그러한 분야의 구체적인 데이터나 지식을 포함하지 않기 때문에 그 분야의 질문에 대한 답변의 정확도가 떨어질 수 있다. 이를 보완하기 위해서는 챗봇이 해당 도메인에 특화된 데이터를 학습하거나, 추가적인 정보 검색 및 연동 과정을 통해 특정 분야에 대한 깊이 있는 답변을 생성할 수 있어야 한다.

최근 몇 년간 온라인 플랫폼을 통한 중고차 시장은 빠르게 성장하고 있다. 경제적 불확실성, 신차 가격 상승, 그리고 온라인 플랫폼의 발전 등의 이유로 많은 소비자들이 신차 대신 중고차를 온라인 플랫폼을 통해 선택하는 경향이 증가하고 있다. 하지만 다양한 중고차 매매 플랫폼 속에서 소비자들은 수많은 옵션 중 자신에게 맞는 차량을 찾기 위해 많은 시간을 소비하고 있다. 특히 자동차에 대한 지식이 부족한 초보 구매자들은 차량의 상태, 가격, 성능 등을 종합적으로 고려하기 어려워 부적절한 차량을 선택할 위험이 있다. 따라서 이러한 문제를 해결하기 위해서는 소비자에게 맞춤형 정보를 제공하는 시스템이 필수적이다.

따라서 RAG(Retrieval-Augmented Generation) 기법을 활용하여 중고차 정보 제공 및 추천 챗봇을 개발하고자 한다. 이 챗봇은 중고차 관련 정보에 대해 실시간으로 답변할 수 있으며, 다양한 블로그와 뉴스에서 수집된 정보를 반영하여 신뢰성 높은 정보를 제공할 수 있다. 이를 통해 사용자들은 중고차 구매 시 반드시 확인해야 할 중요한 정보를 쉽게 확인하고, 더 나은 구매 결정을 내릴 수 있을 것으로 기대된다.

1.2. 연구 목표

RAG(Retrieval-Augmented Generation) 기법을 활용하여 사용자에게 중고차 정보를 제공하고, 사용자가 제시하는 조건에 맞는 매물을 추천하는 챗봇 서비스를 개발하는 것이다. 중고차 관련 문서, 뉴스, 블로그 정보와 실제 중고차 매물 데이터를 크롤링하여 수집한 정보를 기반으로, LLM(Large Language Model)과 결합하여 보다 신뢰성 높은 답변과 추천을 제공하는 모델을 구축한다.

이 모델은 사용자 질문의 맥락을 이해하고, 적합한 차량 정보를 검색하여 추천할 수 있는 기능을 갖추게 된다. 최종적으로 이 챗봇 서비스는 웹 어플리케이션 형태로 제공되어, 사용자들이 중고차 관련 정보를 손쉽게 얻고, 원하는 조건에 맞는 중고차를 추천받는 등의 서비스를 이용할 수 있도록 한다.

2. 연구 배경

2.1. 배경 지식

2.1.1. LLM(Large Language Model)

대규모 언어 모델 LLM(Large Language Model)은 방대한 데이터를 학습하여 자연어와 기타 콘텐츠를 이해하고 생성할 수 있는 모델로, 생성형 AI의 대중화와 다양한 비즈니스 기능에 AI를 통합하는 데 중요한 역할을 하고 있다. 많은 기업이 자연어 이해(NLU) 및 자연어 처리(NLP) 기술을 강화하기 위해 LLM을 도입하고 있으며, 이는 머신러닝, 알고리즘, 신경망, 트랜스포머 모델의 발전과 함께 이루어졌다.

LLM은 Microsoft의 OpenAI ChatGPT-3 및 GPT-4, Meta의 Llama 모델, Google의 BERT, PaLM 모델 등의 예시가 있고 이들은 인터페이스를 통해 대중이 쉽게 접근할 수

있다. 한국에서도 카카오의 KoGPT, 네이버의 HyperCLOVA와 같은 LLM이 개발되어 다양한 자연어 처리 및 생성 작업에 활용되고 있다.

LLM의 작동은 딥러닝 기술과 대량의 텍스트 데이터를 활용하여 이루어진다. 주로 트랜스포머 아키텍처를 기반으로 하여, 여러 계층의 신경망을 통해 이전 단어를 기반으로 다음 단어를 예측하는 방식으로 학습한다. 이 과정에서는 토큰화된 단어를 더 작은 문자 시퀀스로 나누고, 이를 임베딩으로 변환하여 문맥을 숫자로 표현한다. LLM은 수십억 페이지의 텍스트 말뭉치를 통해 학습하며, 문법, 의미론, 개념 관계를 이해하도록 훈련된다.

LLM은 텍스트를 이해하고 생성하는 능력을 바탕으로 문맥에 맞는 답변 제공, 언어 번역, 텍스트 요약, 일반 대화 및 FAQ 응답, 창의적 글쓰기, 코드 생성 등 다양한 기능을 수행한다. 이러한 기능은 수십억 개의 매개변수를 통해 가능하며, 이를 통해 챗봇, 콘텐츠 생성, 연구 지원, 언어 번역 등 다양한 분야에서 새로운 가능성을 열어준다. 예를 들어, 고객 관리에서 사용되는 챗봇이나 블로그 및 마케팅 콘텐츠 생성, 정보 요약, 감성 분석 등의 분야에서 LLM은 업무 효율성을 높이고 고객 경험을 향상시키는 데 기여하고 있다.

2.1.2. RAG(Retrieval-Augmented Generation)

LLM은 딥러닝 모델을 사용하고 대규모 데이터 세트로 학습하여 새로운 콘텐츠를 이해하고 요약하며 생성한다. 그러나 이러한 모델은 학습할 때 사용된 데이터 이후의 정보에 접근할 수 없기 때문에 항상 같은 상태에 머물러 있다. 이로 인해 사용자가 최신 정보를 요구할 때 LLM은 잘못된 응답이나 오래된 답변을 제공하거나, 심지어 존재하지 않는 사실을 만들어내는 할루시네이션 현상이 발생할 수도 있다. 또한, LLM이 관련성 높은 구체적인 응답을 제공하려면 특정 도메인에 맞춰 자신들의 데이터를 바탕으로 답변을 제공하는 모델을 필요로 한다. 예를 들어, 기업은 고객 지원 챗봇을 통해 고객 질문에 대해 회사에 대한 특정 답변을 제공하거나, 내부 데이터에 대한 직원 질문에 답변하는 Q&A 봇을 구축할 수 있다. 그러나 이러한 모델을 구축하는 과정에서 모델을 다시 학습시키지 않고도 효과적으로 활용할 방법이 필요하다.

이런 필요를 충족시키기 위해 RAG(Retrieval-augmented generation)라는 기법이 등장하였다. RAG란 외부 지식 베이스에서 정보를 검색하여 대형 언어 모델(LLM)이 가장 정확하면서도 최신의 정보를 기반으로 하여 답변을 작동하도록 돕는 기법이다. 쉽게 말해, LLM이 답변을 생성할 때 올바르게 최신의 정보를 활용할 수 있게 해주는 도구이다. 예를 들어, RAG는 인터넷이나 다른 데이터베이스와 같은 '외부 지식 저장소'에서 관련 정보를 빠르게 찾아내어, 그 정보를 바탕으로 사용자의 질문에 답변한다. 이렇게 함으로써 LLM은 항상 최신 정보를 제공할 수 있으며, 답변의 출처를 이해하는 데 도움을 준다.

RAG는 간단히 이러한 과정으로 구성된다.

1. 외부 데이터 생성

LLM의 기존 학습 데이터 외에, API, 데이터베이스, 문서 리포지토리 등 다양한 소스에서 새 데이터를 가져온다. 이 데이터는 임베딩 언어 모델을 사용해 수치로 변환하여 벡터 데이터베이스에 저장된다.

2. 관련 정보 검색

사용자의 쿼리는 벡터 표현으로 변환되어 벡터 데이터베이스와 비교한다. 사용자가 특정 질문을 입력하면, 관련된 문서나 데이터를 검색하여 적절한 답변을 제공한다.

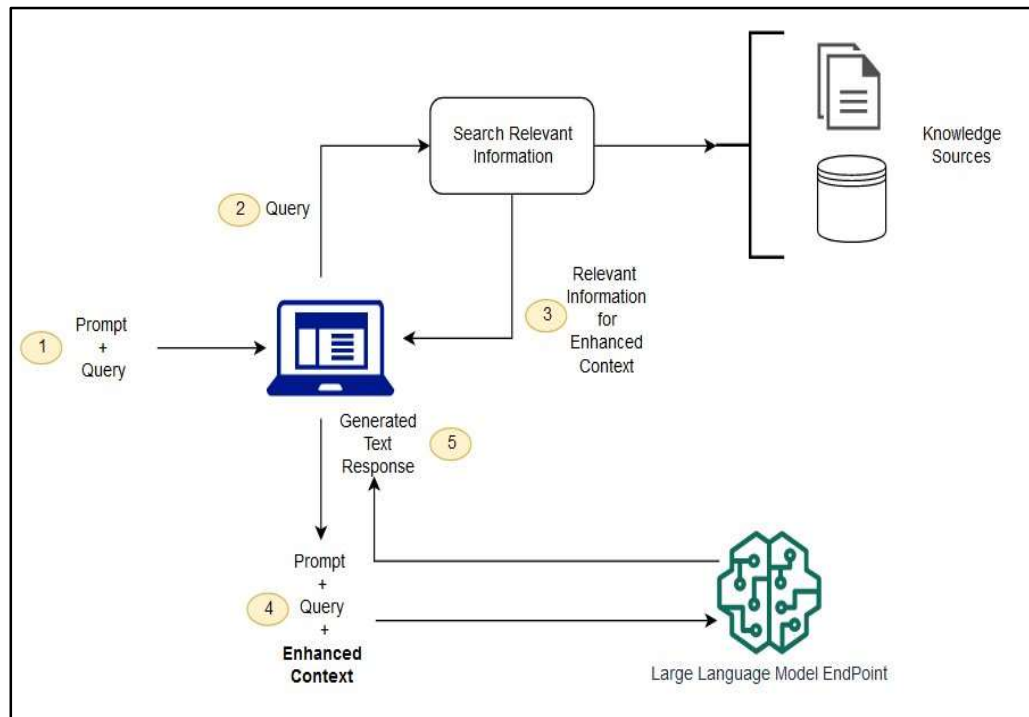
3. LLM 프롬프트 확장

RAG 모델은 검색된 데이터를 사용자 입력에 추가하여 프롬프트를 보강한다. 이렇게 하면 LLM이 더 정확하고 유연한 답변을 생성할 수 있다.

4. 외부 데이터 업데이트

외부 데이터가 최신 정보를 유지하도록 문서를 주기적으로 업데이트하고, 임베딩 표현을 갱신한다. 자동화 실시간 프로세스 또는 주기적 배치 처리로 이 작업을 수행할 수 있다.

다음 다이어그램은 LLM과 함께 RAG를 사용하는 개념적 흐름도이다.

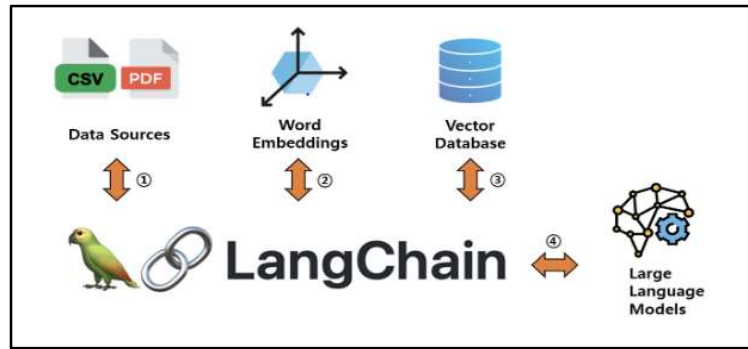


[그림1] RAG 흐름도

파인튜닝된 모델은 특정 시점에서의 데이터로 학습이 완료되기 때문에, 그 시점 이후의 변화하는 정보에는 접근할 수 없다. 또한, 원하는 방향으로의 파인튜닝을 진행하는 것 자체가 쉽지 않으며, 지속적인 재학습과 성능 관리는 어려움을 동반할 수 있다.

이러한 이유로, 변화가 잦은 데이터 환경에서는 파인튜닝 대신 RAG를 사용하는 것이 품질 유지와 비용 측면에서 유리할 수 있다. RAG는 외부 데이터 소스에서 최신 정보를 반영할 수 있으므로, 동적인 환경에서도 효과적으로 활용될 수 있다.

LangChain은 대규모 언어 모델(LLM)을 기반으로 애플리케이션을 개발하기 위한 오픈 소스 오케스트레이션 프레임워크이다. Python과 JavaScript를 지원하는 이 프레임워크는 챗봇, 가상 에이전트 등 LLM을 기반으로 하는 애플리케이션을 구축하는 과정을 간소화한다. LangChain은 다양한 LLM을 위한 일반 인터페이스 역할을 하며, 이를 외부 데이터 소스와 통합할 수 있는 중앙 집중식 개발 환경을 제공한다.



[그림2] Langchain 흐름도

LangChain은 RAG 모델을 쉽게 구축할 수 있도록 다양한 도구와 기능을 제공한다. 이를 통해 개발자는 외부 데이터 소스에서 실시간으로 정보를 검색하고, 해당 정보를 LLM에 통합하여 보다 정확하고 신뢰할 수 있는 응답을 생성할 수 있다. 따라서 LangChain을 활용하면, 데이터의 변화와 사용자 요구에 실시간으로 대응하는 동적인 애플리케이션을 개발할 수 있다.

2.2. 기술 스택

2.2.1. 크롤링

크롤링은 웹 페이지의 데이터를 자동으로 수집하는 기술이다. 주로 Python의 BeautifulSoup, Scrapy, Selenium과 같은 라이브러리를 사용하여 HTML 문서의 구조를 분석하고, 특정 데이터를 추출한다. 이를 통해 뉴스, 상품 정보, 소셜 미디어 데이터 등 다양한 웹 콘텐츠를 자동으로 수집할 수 있다. 크롤링은 데이터 분석, 머신러닝 모델 학습 데이터 수집, 가격 비교 사이트 등에서 활용되며, 주기적으로 업데이트되는 정보를 실시간으로 추적할 때도 유용하다. 검색 엔진의 인덱싱에도 중요한 역할을 하며, 인터넷에서 방대한 양의 데이터를 빠르게 수집하는 데 사용된다.

2.2.2. Flask

Flask는 Python으로 구축된 경량 웹 프레임워크로, 웹 애플리케이션 및 RESTful API 개발에 사용된다. Django와 같은 풀스택 프레임워크에 비해 단순하고 유연한 구조를 갖고 있어, 소규모 프로젝트나 빠른 프로토타입 개발에 적합하다. Flask는 모듈형 구조로, 확장성과 유연성이 뛰어나 필요에 따라 다양한 플러그인을 추가할 수 있다. 기본적

으로 Jinja2 템플릿 엔진과 Werkzeug WSGI 도구를 사용해 웹 페이지의 동적 렌더링과 HTTP 요청 처리를 지원한다. Flask는 데이터베이스와의 통합을 위해 SQLAlchemy와 같은 ORM 라이브러리를 지원하며, 사용자가 원하는 방식으로 쉽게 설정하고 커스터마이징할 수 있다.

2.2.3. Docker

Docker는 애플리케이션을 컨테이너로 패키징하여 운영체제에 독립적인 배포와 실행 환경을 제공하는 오픈 소스 플랫폼이다. 컨테이너는 애플리케이션 실행에 필요한 모든 라이브러리, 종속성, 설정 파일 등을 하나의 독립된 패키지로 묶어 동일한 환경에서 일관되게 실행할 수 있도록 보장한다. 이는 개발 환경과 배포 환경 간의 차이로 인한 문제를 줄이고, DevOps 워크플로우에서 지속적 통합(CI)과 지속적 배포(CD)을 더욱 효율적으로 관리할 수 있게 한다. Docker는 또한 시스템 자원을 매우 효율적으로 사용하며, 여러 컨테이너가 동일한 호스트에서 실행되더라도 각 컨테이너는 고립된 환경에서 동작하기 때문에 충돌 없이 병렬 처리가 가능하다.

2.2.4. 백엔드

1. spring boot

Spring Boot는 자바 기반의 프레임워크로, 스프링 프레임워크를 간편하게 사용할 수 있도록 한다. 설정이 최소화되어 빠른 개발이 가능하며, RESTful API와 마이크로서비스 아키텍처를 쉽게 구축할 수 있다는 장점이 있다. Tomcat과 같은 내장 서버를 지원하여 애플리케이션을 독립적으로 실행할 수 있으며, 다양한 의존성을 통해 필요한 기능을 간편하게 추가할 수 있다. Spring Boot는 보안 기능과 유연한 데이터 접근 방식도 제공하기 때문에 대규모 애플리케이션 개발에 유용하다.

2. node.js

Node.js는 자바스크립트를 서버 측에서 실행할 수 있게 해주는 런타임 환경이다. 비동기 이벤트 기반 구조로 높은 성능을 자랑하며, 많은 동시 연결을 처리할 수 있다. npm(Node Package Manager)을 통해 수많은 라이브러리와 모듈을 쉽고 빠르게 관리할 수 있으며, Express.js와 같은 프레임워크를 사용해 RESTful API 및 웹 애플리케이션

을 효율적으로 개발할 수 있다. 특히나 Node.js는 이번에 개발하게 된 실시간 애플리케이션(예: 채팅 앱, 게임 등)에 유리하다.

3. MariaDB

MariaDB는 MySQL의 오픈 소스 관계형 데이터베이스 관리 시스템(RDBMS)이다. MySQL과 호환되며, 데이터베이스 성능과 안정성을 향상시키기 위한 다양한 기능을 제공한다. ACID(Atomicity, Consistency, Isolation, Durability) 특성을 지원하며, 트랜잭션 처리와 복잡한 쿼리 실행이 가능하다. 특히 데이터베이스 클러스터링, 복제 및 분산 데이터베이스 기능을 통해 대규모 환경에서도 높은 가용성을 유지할 수 있다.

2.2.5. 프론트엔드

1. React.js

React는 사용자 인터페이스를 구축하기 위한 자바스크립트 라이브러리이다. 컴포넌트 기반 아키텍처를 통해 재사용 가능한 UI 구성 요소를 쉽게 만들 수 있으며, 효율적으로 UI 업데이트를 처리할 수 있다. React는 단일 페이지 애플리케이션 개발에 적합하며, 상태 관리 라이브러리(예: Redux, Context API)와 함께 사용하여 복잡한 애플리케이션도 효과적으로 관리할 수 있다. 이러한 특성 덕분에 대규모 애플리케이션에서도 성능과 유연성을 보장한다.

2. typescript

TypeScript는 자바스크립트의 상위 집합으로, 정적 타입을 지원하는 프로그래밍 언어이다. 이를 통해 코드 작성 시 타입을 명시할 수 있어, 오류를 사전에 발견하고 코드의 안정성을 높일 수 있다. TypeScript는 자바스크립트 기능을 지원하고, React, Angular 등의 프레임워크와 잘 통합하여 사용할 수 있다.

3. css

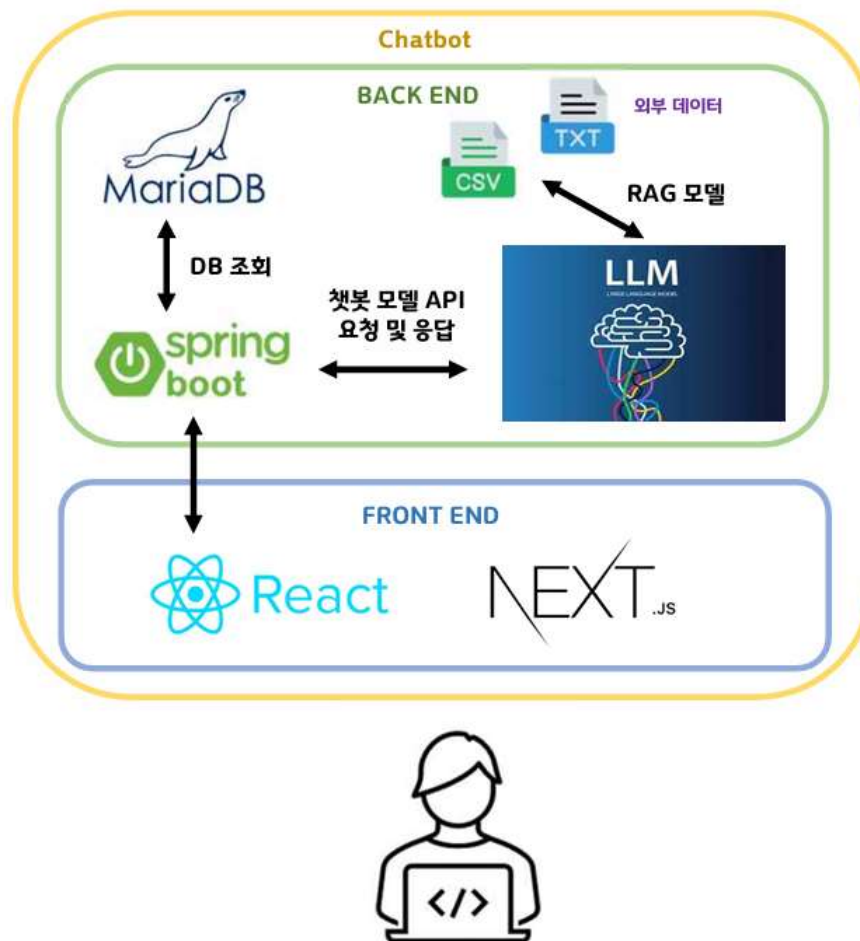
CSS(Cascading Style Sheets)는 웹 페이지의 스타일과 레이아웃을 정의하는 스타일 시트 언어이다. HTML 요소의 색상, 폰트, 여백, 배치 등을 지정할 수 있으며, 다양한 레이아웃 기법을 통해 반응형 디자인을 구현할 수 있다. bootstrap같은 CSS 프레임워크를

사용하면 더 쉽게 디자인을 적용할 수 있으며, CSS 전처리기를 통해 코드의 유지보수성과 재사용성을 높일 수 있다.

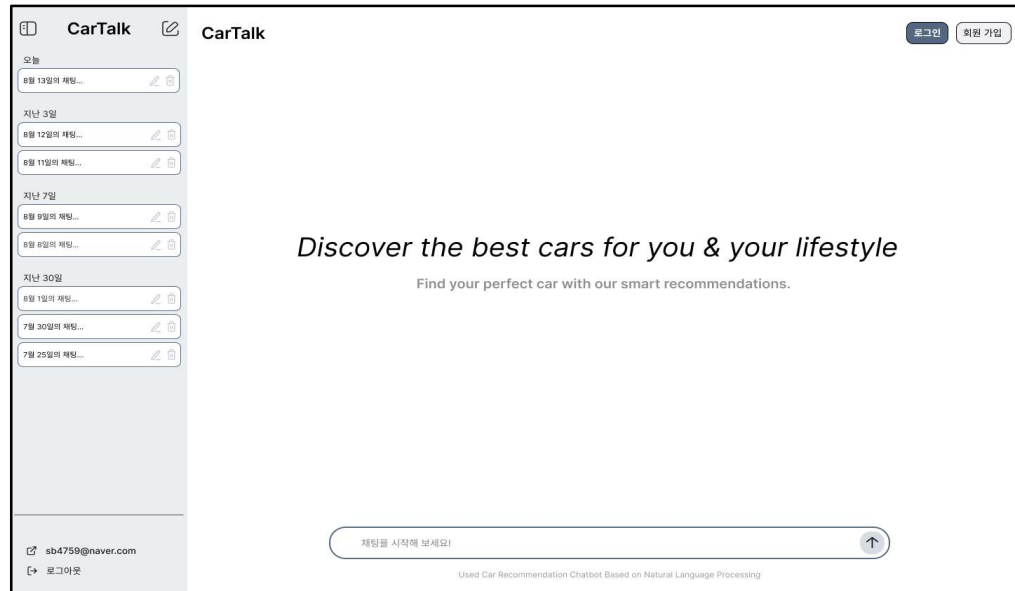
3. 연구 내용

3.1. 챗봇 기본 설계

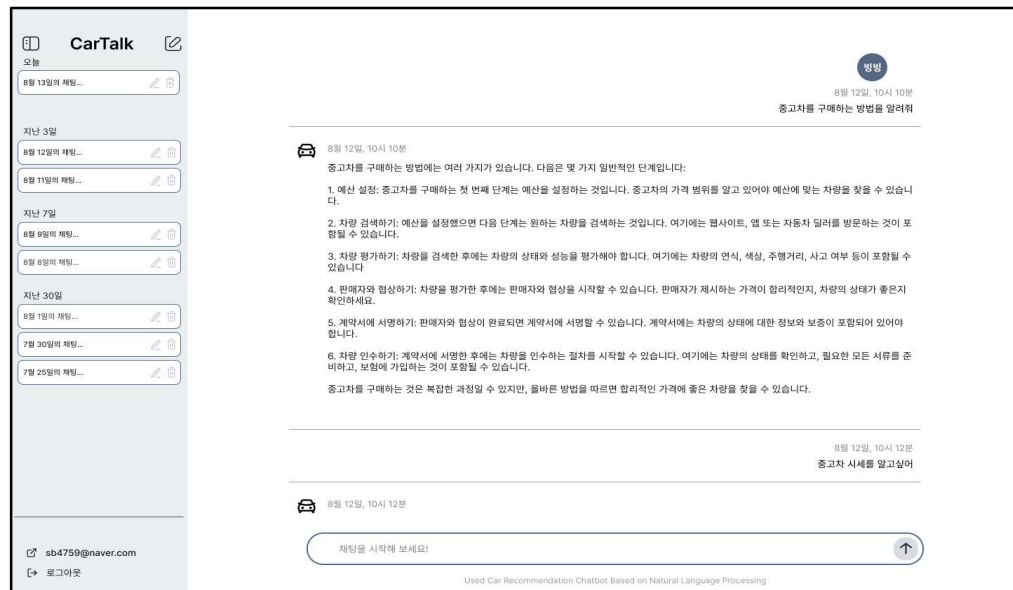
3.1.1. 시스템 구조도



3.1.2. 챗봇 UI 디자인



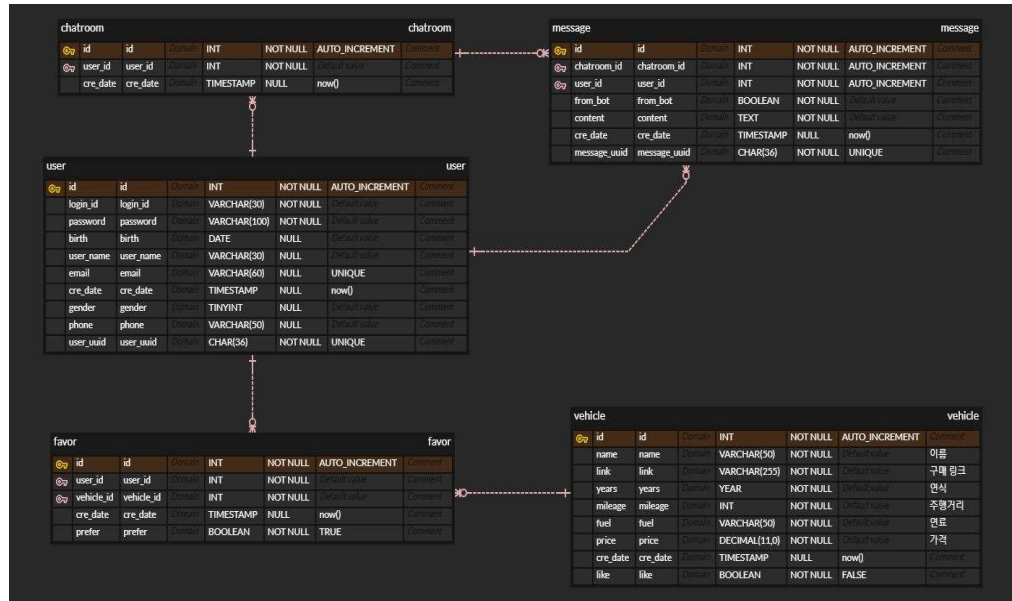
[그림4] 챗봇 main UI - before chat



[그림5] 챗봇 main UI - after chat

Figma를 프로그램을 이용하여 챗봇 UI를 디자인하였다. 챗봇 이름은 CarTalk이며, 사용자가 쉽게 접근하고 대화를 즐길 수 있도록 간단하고 직관적인 인터페이스로 설계했다. 기본적으로 회원가입 및 로그인 기능을 제공하여, 사용자들이 개별 계정을 통해 자신의 대화 기록을 안전하게 관리할 수 있다.

3.1.3. 데이터베이스 설계



[그림6] 데이터베이스

데이터베이스는 챗봇 사용자 정보와 채팅방 정보 및 채팅 내역 등을 관리하는 것을 목표로 하고 있다. 주요 테이블의 역할은 다음과 같다.

- **User(사용자):**
 - 사용자의 로그인 정보(login_id, password), 개인 정보(user_name, email, phone 등) 및 고유한 식별자(user_uuid)를 관리한다.
 - 사용자 정보는 chatroom 테이블과 연결되어 사용자 채팅 기록을 추적한다.
- **Chatroom(채팅방):**
 - 각 채팅방의 고유 아이디(user_id)와 생성 일자(cre_date)를 기록하며, 사용자가 채팅방을 만들 때마다 생성된다.
 - 각 채팅방은 하나 이상의 메시지를 포함하며, message 테이블과 연결된다.
- **Message(메시지):**
 - 채팅방 내에서 주고받은 메시지 정보를 관리하며, 메시지 작성자(user_id) 및 생성 일자(cre_date)를 기록한다.
 - from_bot 필드는 메시지가 챗봇에 의해 전송되었는지 여부를 나타낸다.

3.2. RAG 시스템 구축

3.2.1. 중고차 데이터 크롤링

CarTalk 시스템은 사용자에게 최신 중고차 매물을 추천하는 기능을 제공한다. 이를 위해, 실제 중고차 매물을 웹에서 자동으로 수집하는 크롤링 작업을 수행해야 한다. 크롤링된 데이터는 CSV 파일로 저장되어, RAG 모델을 통해 실시간으로 사용자에게 적합한 중고차 매물을 추천하는 데 활용된다. 크롤링 과정에서는 Python의 다양한 라이브러리를 사용하여 효율적으로 데이터를 수집하고 처리하였다.

- **requests**: HTTP 요청을 보내 웹 페이지의 HTML 소스를 가져오기 위해 사용
- **BeautifulSoup**: 웹 페이지의 HTML을 파싱하고 원하는 정보를 추출하는 데 사용
- **Pandas**: 추출된 데이터를 DataFrame 형식으로 처리하고 CSV 파일 저장시 사용

```
import requests
from bs4 import BeautifulSoup
import pandas as pd

totalurl=[
    ("https://www.bobaedream.co.kr/mycar/mycar_list.php?gubun=I&maker_no=13&group_no=201&page=1&order=S11&view_size=20",1)]

df_cars=[]

urls=[]
for i in totalurl:
    pagenum=i[1]
    for j in range(pagenum):
        url=i[0].format(str(j))
        urls.append(url)

info=["이름", "차량번호", "링크", "연식", "주행거리", "연료", "배기량", "색상", "보증정보", "가격", "신차대비가격"]

coloptions=["옵션 섀루프", "옵션 파노라마섀루프", "옵션 열선앞", "옵션 열선뒤", "옵션 전방센서",
    "옵션 후방센서", "옵션 전방캠", "옵션 후방캠", "옵션 어라운드뷰", "옵션 네비순정"]

findoptions=["섀루프", "파노라마섀루프", "열선시트(앞좌석)", "열선시트(뒷좌석)", "전방센서",
    "후방센서", "전방카메라", "후방카메라", "어라운드뷰", "네비게이션(순정)"]
colacci_info1=["보험이력등록", "소유자변경횟수", "사고상세_전손", "사고상세_침수전손", "사고상세_침수분손", "사고상세_도난", "보험_내차피해"]

cols=info+coloptions+colacci_info1
#링크추가: 아웃라이어인지 확인하기 위해

#옵션이름 받아서 있는지 없는지 확인
def option_check(soupobject,option_name):
    check = soupobject.find("button", text=option_name).find_parent().find_previous_sibling().get_attribute_list('checked')

    if check[0]=='':
        return '유'
    else:
        return '무'
```

[그림7] 크롤링 코드 일부

크롤링을 통해 수집된 데이터는 중고차 매물의 다양한 정보를 포함하고 있다. 차량 이름, 번호, 연식, 주행 거리, 연료, 색상 등의 기본 정보뿐만 아니라 옵션 사항과 보험 이력 정보까지 제공하여, 사용자가 차량을 선택할 때 참고할 수 있는 중요한 요소들을 제

시한다. 또한 다양한 중고차 플랫폼에서 매물을 크롤링함으로써, 사용자는 다양한 출처의 중고차 매물을 추천받을 수 있다.

아래는 전처리를 거친 크롤링 결과 .xlsx 파일의 일부분이다.

	A	B	C	D	E	F	G	
1	브랜드	모델명	이름	연식	주행거리	색상	가격(만원)	
2	현대	스타렉스	그랜드스타렉스 2.5 디젤 밴 3인승 2WD CVX 럭셔리	2011	195981	은색	499	https://www.bobae
3	현대	스타렉스	그랜드스타렉스 2.5 디젤 밴 5인승 2WD CVX 럭셔리	2011	184671	검정색	379	https://www.bobae
4	현대	스타렉스	더 뉴 그랜드스타렉스 2.5 디젤 밴 5인승 모던	2018	101079	남색	1550	https://www.bobae
5	현대	스타렉스	스타렉스 캠핑카	2006	190000	흰색투톤	1399	https://www.bobae
6	현대	스타렉스	그랜드스타렉스 캠핑카	2008	87458	은색	1900	https://www.bobae
7	현대	스타렉스	뉴스타렉스 캠핑카	2005	190000	흰색투톤	1399	https://www.bobae
8	현대	스타렉스	그랜드스타렉스 어린이보호차 디젤 12인승 2WD	2012	147000	노란색	539	https://www.bobae
9	현대	스타렉스	그랜드스타렉스 2.5 디젤 밴 5인승 2WD CVX 프리미엄	2012	154000	은색	529	https://www.bobae
10	현대	스타렉스	더 뉴 그랜드스타렉스 2.5 디젤 밴 3인승 스타일	2018	122286	청색	1250	https://www.bobae
11	현대	스타렉스	그랜드스타렉스 2.5 디젤 밴 3인승 2WD CVX 디럭스	2015	263566	은색	660	https://www.bobae
12	현대	스타렉스	더 뉴 그랜드스타렉스 2.5 디젤 밴 5인승 스마트	2018	73005	은색	1490	https://www.bobae
13	현대	스타렉스	그랜드스타렉스 2.5 디젤 밴 5인승 2WD CVX 프리미엄	2013	253000	은색	550	https://www.bobae
14	현대	스타렉스	더 뉴 그랜드스타렉스 2.5 디젤 밴 3인승 스마트	2018	105326	은색	1360	https://www.bobae
15	현대	스타렉스	그랜드스타렉스 어린이보호차 디젤 12인승 2WD	2015	156190	노란색	649	https://www.bobae
16	현대	스타렉스	그랜드스타렉스 캠핑카	2012	94157	회색	2380	https://www.bobae
17	현대	스타렉스	그랜드스타렉스 2.5 디젤 왜건 12인승 2WD 스마트?	2017	71890	녹색	1390	https://www.bobae
18	현대	스타렉스	그랜드스타렉스 2.5 디젤 밴 3인승 2WD CVX 럭셔리	2013	186668	흰색	770	https://www.bobae
19	현대	스타렉스	그랜드스타렉스 2.5 디젤 왜건 12인승 2WD 스마트	2017	153000	남색	1180	https://www.bobae

[그림8] 크롤링 결과 .xlsx 파일 일부

3.2.2. 중고차 관련 정보 수집

CarTalk 챗봇은 단순히 크롤링 데이터를 바탕으로 중고차 매물을 추천하는 것에 그치지 않고, 중고차 및 차량에 대한 지식이 부족한 사용자들에게도 관련 질문에 대한 답변을 제공하는 것을 목표로 하고 있다. 이를 위해 외부 데이터에서 관련 정보를 검색하고 이를 기반으로 답변하도록 설계하였다. 이를 위해서는 중고차 관련 정보를 검색할 외부 데이터를 수집하는 과정이 필요하다. 중고차 관련 정보를 검색하기 위한 외부 데이터를 다음과 같이 수집하였다.

- **공식 배포 자료:** 한국 소비자원이나 금융 감독원을 비롯한 공식 기관에서 배포하는 중고차 거래 실태 및 중고차 구매 시 주의사항 등의 문서를 수집하였다. 이러한 공식 자료들은 중고차 거래의 현황과 소비자 보호에 대한 귀중한 정보를 제공한다.
- **전문 블로그:** 중고차 관련 전문 블로그에서 유용한 팁과 경험담을 수집하였다. 이러한 글들은 실제 구매자들의 관점과 조언을 반영하여, 사용자가 중고차를 구매하는 데 도움이 되는 실질적인 정보를 제공한다.

- **데이터 로딩:** 중고차 관련 외부 데이터를 담고 있는 텍스트 파일을 `TextLoader`를 사용하여 로드한다.
- **텍스트 분할:** 문서는 크기가 크므로 `RecursiveCharacterTextSplitter`를 이용해 문서를 작은 청크로 나누어 더 효율적으로 정보를 검색하고 처리하도록 한다.

```
from langchain_openai import OpenAIEmbeddings
from langchain_community.vectorstores import FAISS

embeddings = OpenAIEmbeddings()

vectorstore = FAISS.from_documents(documents=split_documents, embedding=embeddings)

retriever = vectorstore.as_retriever()
```

[그림11] 임베딩 및 벡터 저장소 구축

- **임베딩:** 분할된 문서를 기반으로 `OpenAIEmbeddings` 이용해 문서의 의미를 수치적으로 표현하는 임베딩을 생성한다.
- **벡터 저장소 구축:** 생성된 임베딩은 `FAISS` 벡터 스토어에 저장되어, 사용자 질문에 대한 관련 문서를 신속하게 검색할 수 있도록 한다.

```
import pandas as pd

crawling_df = pd.read_csv("/content/crawling_data.CSV", encoding="cp949")
```

[그림12] 크롤링 데이터 로드

- **CSV 파일 로딩:** `pandas`를 이용해 CSV 파일 형식으로 저장된 중고차 매물 크롤링 데이터를 로드한다.

```
from langchain_openai import ChatOpenAI
from langchain.agents.agent_types import AgentType
from langchain_experimental.agents.agent_toolkits import create_pandas_dataframe_agent

agent = create_pandas_dataframe_agent(
    ChatOpenAI(temperature=0, model="gpt-3.5-turbo"),
    crawling_df,
    verbose=True,
    agent_type=AgentType.OPENAI_FUNCTIONS,
    allow_dangerous_code=True
)
```

[그림13] 에이전트 생성

- **에이전트 생성:** `create_pandas_dataframe_agent`을 이용해 로드한 크롤링 데이터를 기반으로 OpenAI 모델과 연결된 에이전트를 생성한다. 이 에이전트는 CSV 데이터에서 질문에 대한 답변을 찾는 역할을 한다.

```
from langchain_core.prompts import PromptTemplate

pdf_prompt = PromptTemplate.from_template(
    """너는 중고차나 자동차 관련 정보를 제공해줘야 해.
    관련 질문에 답변을 하기 위해 아래의 context를 참고하렴.
    답변을 할 때는 그렇게 판단하는 이유도 자세히 포함시켜주면 좋겠어.
    잘 모르는 질문에 대해서는 굳이 답변하지 않아도 돼.
    Answer in Korean.

    #Context:
    {context}

    #Question:
    {question}

    #Answer: """)
```

[그림14] 중고차 정보 관련 질문 프롬프트

```
csv_prompt = PromptTemplate.from_template(
    """너는 조건에 맞는 중고차 매물을 추천해줘야 해.
    중고차 매물을 추천하기 위해 아래의 context를 참고하렴.
    조건에 맞는 매물이 많다면, 3개를 랜덤으로 선택해 실제 링크를 포함한 매물 정보를 제공해줘.
    조건에 맞는 매물이 없을 경우, 조건에 맞는 매물이 없으니 다른 조건을 제시해달라고 요청해줘.
    Answer in Korean.

    #Context:
    {context}

    #Question:
    {question}

    #Answer:
    """)
```

[그림15] 중고차 매물 관련 질문 프롬프트

- **프롬프트 생성:** 모델이 사용자 질문에 대해 어떻게 응답해야 하는지 프롬프트를 정의한다. 각각 중고차 정보 관련 질문에 대한 프롬프트와 중고차 매물 추천 질문에 대한 프롬프트이다. 프롬프트는 언어 모델에게 질문에 대한 답변을 생성하는 데 필요한 지침을 제공한다. `context`에는 외부 데이터에서 검색한 내용이 포함되어 언어 모델이 답변을 생성할 때 참고할 수 있도록 한다.

```

def get_response(question):
    if "매물" in question and "추천" in question:
        chain = (
            {"context": agent.invoke, "question": RunnablePassthrough()}
            | csv_prompt
            | llm
            | StrOutputParser()
        )
    else:
        chain = (
            {"context": retriever, "question": RunnablePassthrough()}
            | pdf_prompt
            | llm
            | StrOutputParser()
        )

    response = chain.invoke(question)
    return response

```

[그림16] 질문 처리 함수

- **질문 처리 함수:** 사용자의 질문을 기반으로 적절한 체인을 선택하여 응답을 생성한다. 질문에 "매물" 및 "추천"이라는 키워드가 포함되어 있으면 중고차 매물 추천 체인을 사용하고, 그렇지 않으면 일반 정보 제공 체인을 사용한다. 이러한 구조는 각 질문의 성격에 따라 최적의 응답을 생성할 수 있도록 한다.

```

question = "제네시스 G70 모델 중 연식이 2020년 이후인 매물을 추천해줘"
response = get_response(question)
print(response)

```

[그림17] 중고차 매물 추천 질문

```

> Finished chain.
제네시스 G70 모델 중 연식이 2020년 이후인 매물을 추천해드립니다:
1. **더 뉴 G70 2.0 터보 스포츠 팩**
- 연식: 2021
- 주행거리: 35,094km
- 색상: 흰색
- 가격: 3,890만원
- 링크: [매물 보러가기] (https://www.bobaedream.co.kr/mycar/mycar\_view.php?no=매물번호&gubun=K)

2. **더 뉴 G70 슈팅브레이크 2.0 터보 AWD 스포츠 팩**
- 연식: 2023
- 주행거리: 2,356km
- 색상: 녹색
- 가격: 4,499만원
- 링크: [매물 보러가기] (https://www.bobaedream.co.kr/mycar/mycar\_view.php?no=매물번호&gubun=K)

3. **더 뉴 G70 2.0 터보 스포츠 팩**
- 연식: 2021
- 주행거리: 14,967km
- 색상: 검정색
- 가격: 3,700만원
- 링크: [매물 보러가기] (https://www.bobaedream.co.kr/mycar/mycar\_view.php?no=매물번호&gubun=K)

```

[그림18] 중고차 매물 추천 답변

중고차 매물 추천과 관련된 질문의 경우, CSV 파일에서 조건에 부합하는 매물 데이터를 검색하고, 이를 바탕으로 언어 모델이 답변을 생성한다.

질문	중고차를 구입하려고 하는데 가장 믿을만한 중고차 구매 플랫폼을 알려줘.
기존 모델	<p>중고차를 구매하려는 경우, 다음과 같은 중고차 구매 플랫폼을 추천해 드립니다.</p> <p>1.중고나라: 국내 최대 중고거래 플랫폼으로 다양한 중고차 정보를 제공하고 안전한 거래를 보장합니다.</p> <p>2.오토벨: 전문적인 중고차 거래 플랫폼으로 신뢰할 수 있는 판매자와 구매자를 연결해줍니다.</p> <p>3.카매니아: 다양한 브랜드와 모델의 중고차 정보를 제공하며 안전한 거래를 지원합니다.</p> <p>4.중고차마켓: 중고차 거래 전문 플랫폼으로 다양한 차량 정보와 실제 구매자 후기를 확인할 수 있습니다.</p> <p>이러한 중고차 구매 플랫폼을 통해 신뢰할 수 있는 정보를 확인하고 안전하게 중고차를 구매할 수 있습니다.</p>
RAG 모델	<p>엔카가 가장 믿을만한 중고차 구매 플랫폼으로 추천됩니다. 최근 조사 결과에 따르면, 엔카는 플랫폼을 이용한 중고차 구입자 중 알거나 들어본 적이 있는 비율이 85%에 달하며, 구입 과정에 이용해 본 비율은 73%로 나타났습니다. 또한, 엔카에서 구입한 비율도 60%로 절반을 훌쩍 넘는 것으로 나타났습니다. 이러한 결과를 통해 엔카가 국내 1위 중고차 플랫폼으로서 신뢰받고 있는 것을 확인할 수 있습니다. 또한, 엔카는 다양한 정보와 신뢰를 바탕으로 소비자들의 만족도를 높이는 데 기여하고 있습니다. 따라서 중고차를 구입하려는 경우, 엔카를 고려해보시는 것이 좋을 것입니다.</p>
질문	내 차를 중고차로 판매하려고 해. 어떤 플랫폼을 이용하는게 좋을까?
기존 모델	<p>중고차를 판매하려면 온라인 중고차 거래 플랫폼을 활용하는 것이 좋습니다. 대표적인 중고차 거래 플랫폼으로는 중고나라, 번개장터, 중고차마켓, 카톡중고나라, 중고차나라 등이 있습니다. 이러한 플랫폼을 통해 중고차를 판매하면 더 많은 구매자들에게 노출되어 빠르고 효율적으로 거래를 진행할 수 있습니다. 또한, 중고차 전문 플랫폼인 오토벨, 카매니아, 카페24 등을 활용하여 전문적인 중고차 거래를 할 수도 있습니다.</p>

RAG 모델	중고차를 처분하려는 경우, 헤이딜러나 AJ셀카, 엔카, KB차차차와 같은 중고차 플랫폼을 이용하는 것이 좋습니다. 이는 소비자들의 만족도가 높고, 플랫폼에 대한 신뢰와 다양한 정보를 제공하기 때문입니다. 특히 헤이딜러는 '내차팔기'를 앞세운 선택과 집중 전략, 톱모델을 기용한 과감한 마케팅으로 인지도를 높였고, 딜러 관리도 매우 엄격하여 소비자의 신뢰를 확보한 것으로 알려져 있습니다. 따라서 이러한 중고차 플랫폼을 활용하여 중고차를 처분하는 것이 좋은 선택일 것입니다.
질문	프리미엄 차를 구입하려고 해. 제네시스랑 벤츠 중에 뭐가 더 좋을까?
기존 모델	두 차 모두 프리미엄 브랜드이며 높은 품질과 성능을 자랑합니다. 그러나 제네시스는 비교적 저렴한 가격에 높은 품질을 제공하는 것으로 유명하며, 최근에는 많은 수상을 받으며 주목을 받고 있습니다. 반면에 벤츠는 오랜 역사와 전통을 자랑하며 뛰어난 성능과 고급스러운 디자인으로 유명합니다. 따라서 어떤 차를 선택할지는 개인의 취향과 우선순위에 따라 다를 수 있습니다. 제네시스는 비교적 저렴한 가격에 높은 품질을 원하시는 분들에게 적합할 수 있고, 벤츠는 전통과 고급스러운 이미지를 중요시하는 분들에게 더 적합할 수 있습니다. 두 차를 직접 비교해보고 자신에게 가장 적합한 차를 선택하는 것이 좋을 것입니다.
RAG 모델	제네시스와 벤츠 중에서 선택할 때, 최근 연례 자동차 기획조사 결과를 보면 프리미엄차 구입자들이 가장 많이 제네시스와 벤츠를 비교한 후 제네시스를 선택한 경향이 있습니다. 또한, 제네시스는 구입 시점에서 경쟁 규모 톱5 중 4개 올라가는 등 경쟁력을 강화하고 있습니다. 브랜드 간 경쟁 구도에서도 제네시스가 우세한 모습을 보이고 있습니다. 따라서, 현재 제네시스가 프리미엄 차 시장에서 높은 평가를 받고 있으며, 선택 시에는 제네시스를 고려해보는 것이 좋을 것으로 판단됩니다.
질문	괜찮은 네비게이션 어플을 추천해줘.
기존 모델	구글 지도(Google Maps)를 추천합니다. 실시간 교통 정보, 경로 안내, 건물 내부 지도, 식당 및 가게 정보 등 다양한 기능을 제공하며 정확한 길 안내로 많은 사용자에게 신뢰를 받고 있습니다. 또한 사용이 간편하고 무료로 이용할 수 있어 많은 사람들이 선호하는 어플 중 하나입니다.

RAG 모델	티맵이 내비게이션 앱 중에서 가장 많은 이용자를 보유하고 있습니다. 조사 결과에 따르면 스마트폰 내비 이용자 중 74%가 티맵을 주로 사용하고 있습니다. 티맵은 길 안내가 빠르고 정확하다는 이유로 많은 이용자들에게 선호를 받고 있습니다. 또한, 티맵의 이용률은 국산차와 수입차 모두에서 높은 편이며, 최근에는 수입차에 기본 장착되는 경우도 늘어나고 있습니다. 따라서 티맵을 추천드립니다.
질문	연식이 변경된 후인 연초에 중고차를 구입하면 가격이 저렴할까?
기존 모델	일반적으로 연식이 변경된 후인 연초에 중고차를 구입하면 가격이 상대적으로 저렴할 수 있습니다. 이는 새로운 연식의 차량이 출시되면 기존 연식의 차량이 가치가 하락하기 때문입니다. 또한 새로운 연식의 차량이 시장에 나오면 중고차 시장에 중고차가 많이 판매되기 때문에 가격이 조정될 수 있습니다. 그러나 중고차의 가격은 연식뿐만 아니라 차량의 상태, 주행거리, 유지보수 이력 등 여러 가지 요인에 따라 다를 수 있으므로 구매 전에 충분한 조사와 비교 분석이 필요합니다. 또한 중고차를 구입할 때는 신중하게 차량을 검사하고 중고차의 역사를 확인하는 것이 중요합니다.
RAG 모델	연식이 변경된 후인 연초에 중고차를 구입하면 가격이 더 저렴해진다고 일반적으로 생각할 수 있지만, 실제 시장에서는 그렇지 않을 수 있습니다. 연식변경에 따른 가격 하락은 이미 연말부터 시작되며, 중고차 시장에서는 연식에 대한 감가가 이미 충분히 이뤄져 있는 경우가 많습니다. 또한, 봄 시즌에는 중고차 성수기로 인해 연말에 비해 더 저렴하게 거래되지 않을 수도 있습니다. 따라서 연초에 중고차를 구입할 때는 연식변경에 따른 가격 하락을 기대하기보다는 시장 상황과 거래 패턴을 고려하여 구매 결정을 하는 것이 좋습니다.

[표1] 중고차 관련 질문 및 답변

기존 모델도 이미 광범위한 데이터에 기반하여 상당히 유의미한 답변을 제공하고 있다. 예를 들어, 중고차 구매 플랫폼을 추천할 때, 기존 모델은 중고나라, 오토벨 등 다양한 신뢰할 만한 플랫폼을 제시하며, 구체적인 장점을 설명한다. 이러한 답변은 충분히 신뢰할 수 있고, 많은 사용자들에게 유용할 수 있다.

그러나 RAG 모델은 외부 데이터에서 최신 정보를 참고하여 답변을 제시하기 때문에 기존 모델보다 좀 더 현실적이고 최신 상황을 반영하는 답변을 제시한다. 예를 들어, '엔카'를 추천할 때, 최근 조사 결과를 인용하며 실제 구입자들의 비율이나 만족도를 반영해 더 구체적이고 신뢰성 있는 답변을 제공한다. 이처럼 RAG 모델은 최신 정보에 기반한 현실적인 답변을 제공하여, 사용자가 더 나은 결정을 내릴 수 있도록 도와준다.

3.3. 모델 API 구현

3.3.1. Flask

Flask를 이용하여 RESTful API를 제작하였다. 이 API는 사용자의 질문을 받아 HTTP POST 요청을 처리하고, RAG 모델을 통해 답변을 생성한 후, 그 결과를 JSON 형식으로 반환한다.

```
app = Flask(__name__)

@app.route('/chatbot', methods=['POST'])
def chatbot():
    data = request.get_json()
    question = data.get("question")

    if not question:
        return jsonify({"error": "No question provided"}), 400

    response = get_response(question)
    return jsonify({"response": response})

if __name__ == "__main__":
    app.run(port=5000)
```

[그림19] Flask

`@app.route` 데코레이터는 Flask에서 특정 URL 경로에 대한 요청을 처리하기 위해 사용된다. 여기서는 `/chatbot` 경로에 대한 요청을 설정한다.

`methods=['POST']`는 이 엔드포인트가 HTTP POST 요청만 처리하도록 설정한다. POST 요청은 일반적으로 클라이언트가 서버로 데이터를 전송할 때 사용되며, 이 경우에는 사용자가 챗봇에게 질문을 보낼 때 사용된다.

`app.run(port=5000)`은 Flask 애플리케이션을 실행하는 코드이다. 애플리케이션이 로컬 서버에서 실행되도록 ngrok을 사용해서 터널링한다. 포트 번호 5000으로 대기하며 사용자는 이 포트로 API 요청을 보낼 수 있다.

3.3.2. Docker

앞서 제작한 Flask 애플리케이션이 다양한 환경에서 일관되게 실행되고, 배포 및 관리를 간편하게 할 수 있도록 Docker 컨테이너로 제작하였다. 이를 통해 Flask 애플리케이션은 운영체제와 관계없이 동일한 환경에서 실행될 수 있으며, 의존성 문제도 해결할 수 있다.

```
# Dockerfile
FROM python:3.9-slim

WORKDIR /app

# 필요한 패키지 설치
COPY requirements.txt ./
RUN pip install --no-cache-dir -r requirements.txt

# 애플리케이션 코드 복사
COPY . .

# 서버 실행
CMD ["python", "app.py"]
```

[그림20] Dockerfile 파일

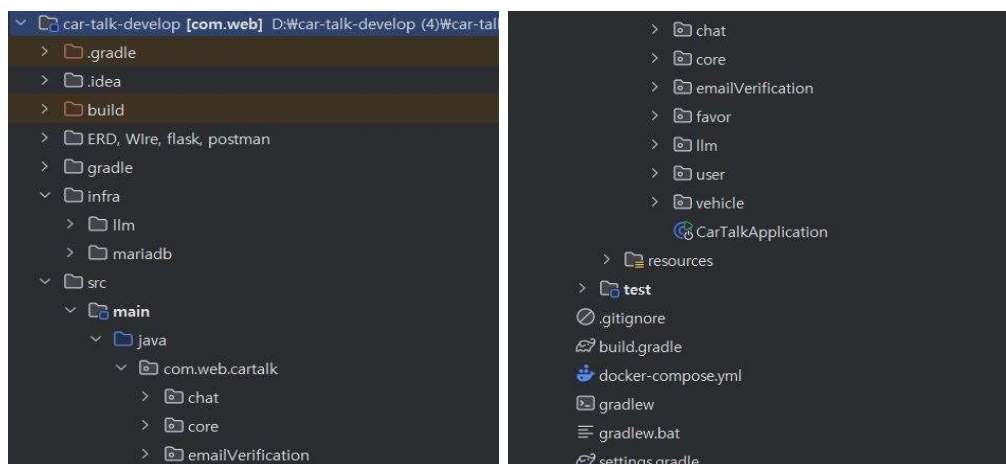
```
flask
redis
pandas
langchain
langchain-openai
langchain-community
```

[그림21] requirements.txt 파일

requirements.txt 파일을 컨테이너의 /app 디렉토리에 복사하고, 해당 파일에 정의된 Python 라이브러리를 설치한다. 일반적으로 requirements.txt 파일에는 app.py 실행에 필요한 라이브러리와 버전 정보가 담겨 있다.

3.4. 백엔드 개발

3.4.1. 패키지 재구성 및 수정



[그림22] 패키지 재구성

사용하지 않는 기능, 수정한 파일들에 대한 패키지를 재구성하였다.

1) websocket 기능 삭제

websocket은 클라이언트와 서버 간에 양방향 통신이 가능해져, 메시지를 즉각적으로 주고받을 수 있다는 장점이 있다. 다만, LLM의 경우 서비스의 흐름이 사용자 -> 서버 -> LLM -> 서버 -> 사용자로 구성되어 있다. 이 부분에서 LLM에서 답변을 생성하고 사용자에게 전달하기까지 대기 시간이 존재하기 때문에 websocket의 필요성이 감소하였고 HTTP 통신을 이용하기로 하였다.

2) SMTP 설정

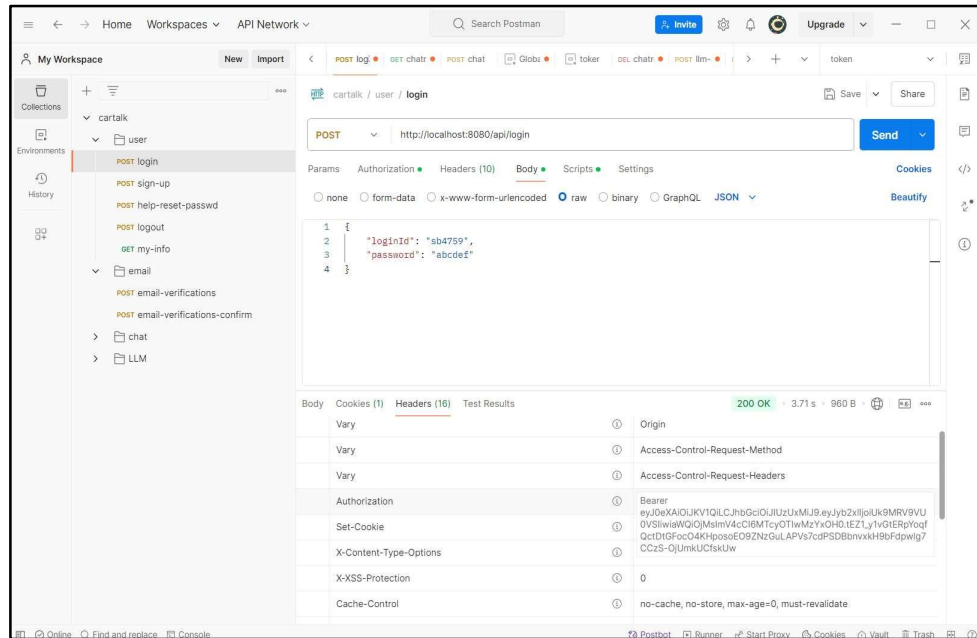
email_verification(이메일 인증), reset password(임시 비밀번호 발급)을 위한 SMTP 설정을 진행했다. 이때, 메일을 전송하기 위한 google 계정을 생성하였다. api 테스트를 통해 정상적으로 동작함을 확인하였다.

3) docker 파일 생성

app.py 등 다른 환경에서 동시에 작업해야하는 경우, 운영 체제에 독립적이기 때문에 개발 환경, 테스트 환경, 프로덕션 환경 간에 쉽게 이동할 수 있다. pip를 사용해 강제로 설치해야하는 상황에서 여러 파일을 설치하는 과정을 피할 수 있고 병렬적으로 작업해 부담을 줄여준다.

3.4.2. API 테스트

docker에서 필요한 elasticsearch, redis, llm, mariadb를 동작시키고, intellij에서 서버를 동작시켜 테스트 세팅을 하고, postman으로 api 테스트를 진행했다. 이 중에서 로그인에 대한 api만 설명하도록 하겠다. 채팅방과 관련된 기능을 사용하기 위해서는 회원가입 후 이메일 인증을 통해 user의 role을 ROLE_USER로 변경한다. 이후 로그인을 진행하면 access Token과 refresh Token을 발급받음을 확인할 수 있다.



[그림23] 로그인 후 access token

localhost:8080/h2-console로 접속해 확인했을 때 user의 정보가 저장되어 있음을 확인할 수 있었다.

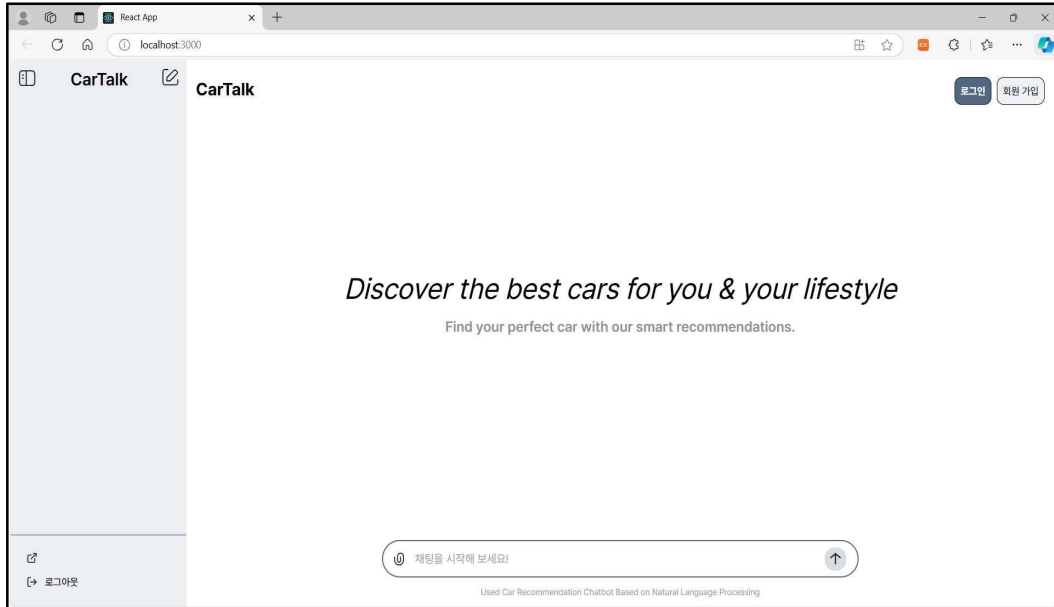


[그림24] h2 database(현재 사진은 더미데이터)

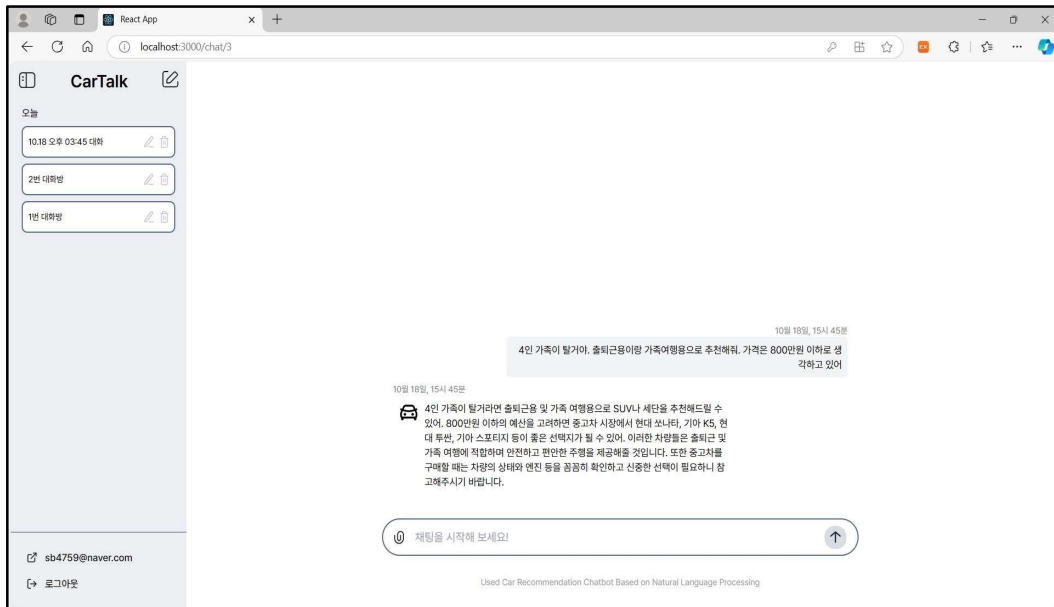
이와 동일하게 다른 api 테스트를 진행하였고 h2 DB에 저장됨을 확인하였다.

3.5. 프론트엔드 개발

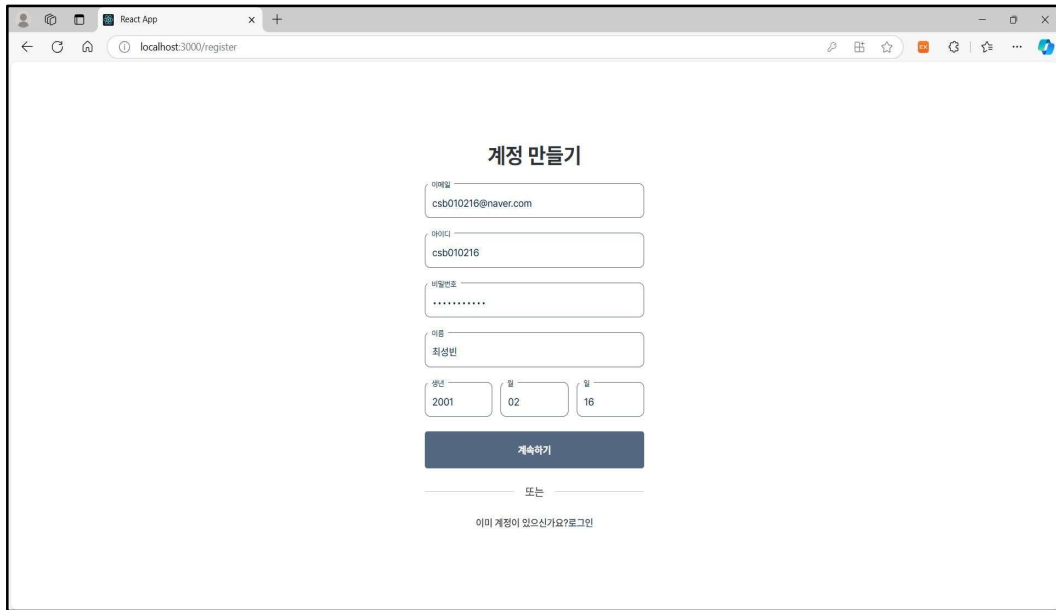
Next.js, TypeScript를 사용하여 채팅 웹 UI를 제작하였으며 개발한 페이지로는 메인 화면, 로그인, 회원가입, 이메일 인증, 비밀번호 초기화(안내문)가 있다.



[그림25] 메인 페이지(로그인 전)



[그림26] 메인 페이지(로그인 후)



계정 만들기

이메일
csb010216@naver.com

아이디
csb010216

비밀번호

이름
최성민

생년
2001

월
02

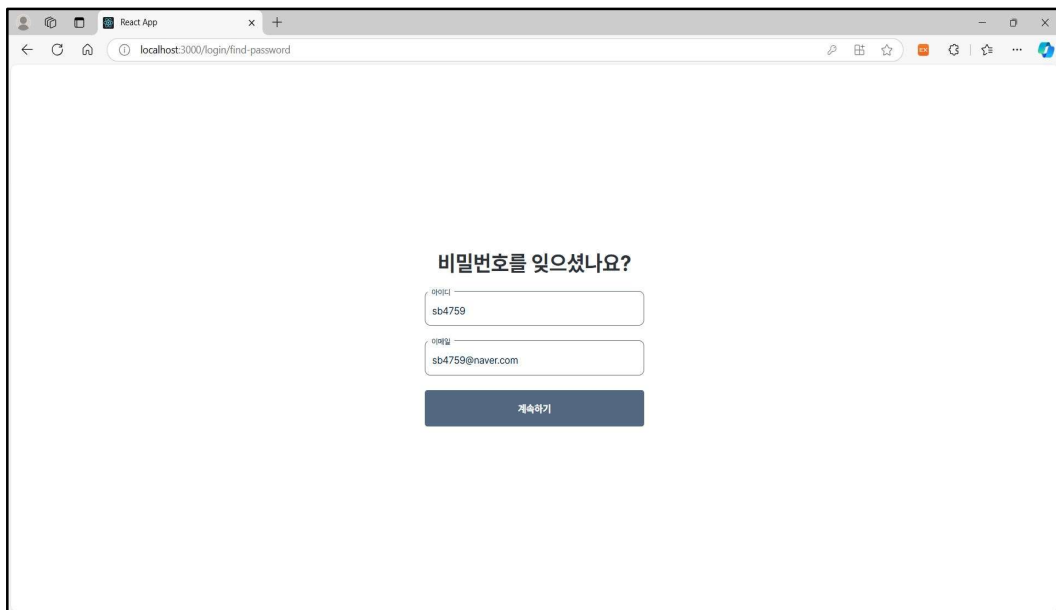
일
16

계속하기

또는

이미 계정이 있으신가요? 로그인

[그림27] 회원가입 페이지



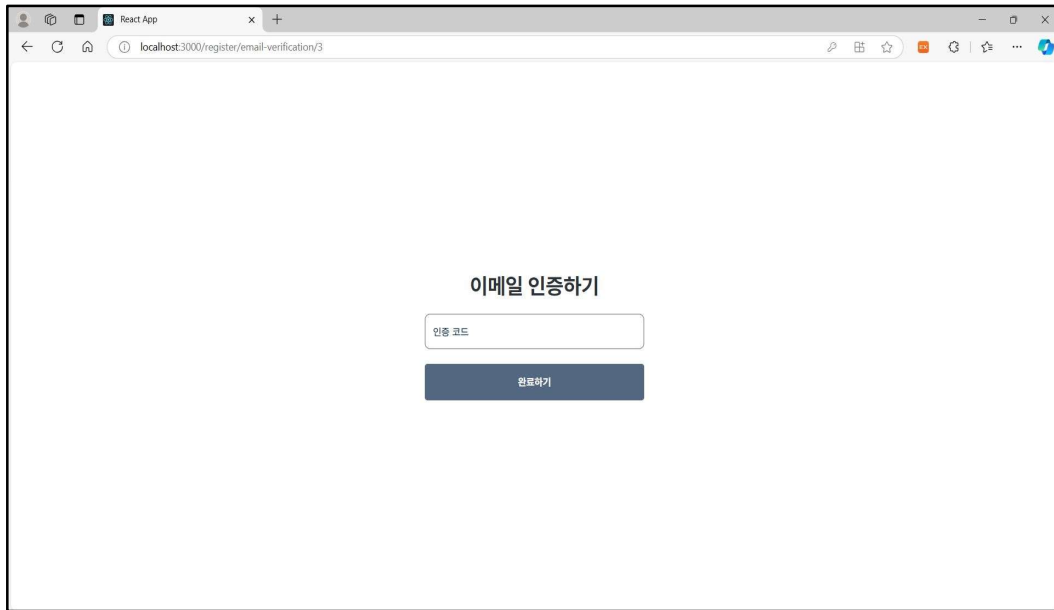
비밀번호를 잊으셨나요?

아이디
sb4759

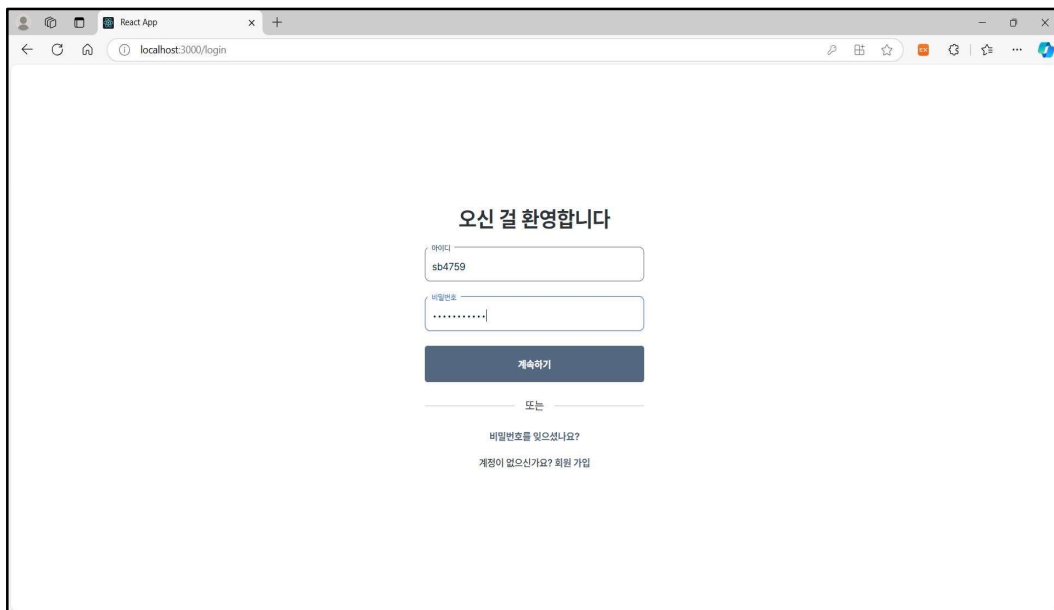
이메일
sb4759@naver.com

계속하기

[그림28] 비밀번호 초기화 페이지



[그림29] 이메일 인증 페이지



[그림30] 로그인 페이지

4. 연구 결과 분석 및 평가

이번 연구는 LLM을 활용한 중고차 매물 추천 및 정보 제공 챗봇을 개발하여 사용자들이 쉽게 중고차 매물을 추천받고 관련 정보를 얻을 수 있도록 하는 것을 목표로 하였다. 특히, RAG(Recovery-Augmented Generation) 기법을 활용하여 외부 데이터에 기반한 신뢰성 높은 답변을 제공하는 데 중점을 두었다.

챗봇 서비스는 웹 어플리케이션을 통해 제공되며, 사용자들이 친숙하게 느낄 수 있도록 GPT와 유사한 디자인으로 설계되었다. 또한, 로그인 기능을 통해 개인별 채팅 기록을 보관하여 사용자의 편의성을 증대시켰다.

그러나 현재 RAG의 성능에는 여전히 개선의 여지가 있다. 외부 데이터를 활용하여 최신 정보를 제공하는 것은 가능하지만, 현재 사용되는 외부 데이터가 주로 뉴스 기사와 블로그 글로 제한되어 있어 정보의 다양성과 깊이가 부족하다. 따라서 커뮤니티 반응이나 사용자 리뷰와 같은 다양한 유형의 데이터를 언어 모델에 추가적으로 결합하여 챗봇의 성능을 향상시킬 필요가 있다.

5. 결론 및 향후 연구 방향

챗봇의 성능을 더욱 향상시키기 위해, 다음과 같은 몇 가지 발전 방안을 고려할 수 있다. 이를 통해 보다 정교한 맞춤형 추천 시스템을 제공하고, 사용자의 편의를 더욱 높일 수 있을 것이다.

- **실시간 중고차 매물 업데이트:** 중고차 매물을 실시간으로 업데이트할 수 있는 방안을 마련할 예정이다. 중고차 거래 플랫폼의 API를 연동할 가능성을 검토하고,, 이를 통해 최신 매물 정보를 사용자에게 제공할 수 있도록 할 계획이다.
- **피드백 기반 맞춤형 추천 시스템:** 사용자가 추천받은 매물에 대해 '좋아요' 또는 '싫어요'와 같은 피드백을 제공할 수 있는 기능을 추가할 예정이다. 이러한 피드백을 바탕으로 더 개인화된 추천을 제공하며, 사용자가 '좋아요'를 선택한 매물과 유사한 매물을 추천하고, '싫어요'를 선택한 매물과 다른 특성을 가진 매물을 우선 추천하는 방식이다. 이와 같은 맞춤형 시스템은 사용자 만족도를 높일 수 있을 것이다.

-
- **차량 시각적 정보 제공:** 챗봇에서 추천된 차량의 외관 사진을 함께 제공하는 시각적 기능을 추가할 계획이다. 이를 통해 사용자는 차량의 상태를 보다 직관적으로 파악할 수 있으며, 매물의 신뢰성을 더욱 높일 수 있을 것이다.
 - **지역 기반 매물 추천:** 사용자가 위치한 지역을 기반으로 매물을 추천하는 기능을 추가할 계획이다. 중고차 구매 시 가까운 지역에서 거래하는 것이 편리하고 비용 절감에도 도움이 될 수 있기 때문에, 사용자의 현재 위치 정보나 선호하는 지역을 반영하여 그 지역 내에서 매물을 추천할 수 있는 시스템을 구축하려고 한다.
 - **정보 제공의 다양성 향상:** 정보 제공에 있어서도 실제 사용자 리뷰나 커뮤니티 반응과 같은 다양한 출처의 데이터를 모델에 적용하여, 보다 심층적이고 신뢰할 수 있는 정보를 사용자에게 제공하는 데 힘쓸 것이다.
 - **DB 필드 수정 및 api 추가 개발:** 특히 vehicle 테이블의 필드 수가 초기의 설계보다 적어졌다. 필드 수가 과하게 많아지면 LLM의 답변이 구체적으로 생성될 수 있지만, 서비스 자체의 성능이 저하될 우려가 있다. 추가적인 필드가 있을 경우 적은 수의 데이터를 추가할 수 있게 수정할 예정이다. 또한 favor 테이블의 기능인 선호도 표시 기능이 미구현 되어있다. 매물의 정보를 정적으로 대입하기 때문에 시간이 흘러 매물이 거래되거나, 정보가 폐기되었을 때 실제로 사용되지 않는 더미데이터로 남아있게 될 수 있어 구현하지 않았다. 향후 업데이트 계획이 세워진다면 추가할 예정이다.

6. 구성원별 역할 및 개발 일정

6.1. 구성원별 역할

이름	역할
이은진	<ul style="list-style-type: none">• 챗봇 웹페이지 UI 디자인 설계• 중고차 관련 외부 데이터 수집• RAG 시스템 설계 및 모델 구현• Flask를 이용한 챗봇 API 구축• 프론트엔드 일부 기능 개발
최희웅	<ul style="list-style-type: none">• 중고차 매물 크롤링 및 전처리• 중고차 관련 외부 데이터 수집• Flask를 이용한 챗봇 API 구축• 프론트엔드 일부 기능 개발
최성빈	<ul style="list-style-type: none">• 데이터베이스 구조 설계• 챗봇 API Docker 컨테이너화• 백엔드 전체 시스템 개발• 프론트엔드 일부 기능 개발• 버그 수정 및 최종 테스트

[표2] 구성원별 역할

6.2. 개발 일정

7월					8월					9월					10월	
1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2
언어 모델 선정 및 테스트																
		DB 설계														
			UI 디자인													
				백엔드 환경 구축												
				데이터 크롤링 및 전처리												
				중고차 관련 데 이터 수집												
					중간 보고서											
							RAG 모델 구현									
										백엔드 개발						
													프론트 개발			
													테스트 및 수정			
														최종 보고서 작 성 및 시연 준비		

[표3] 개발 일정

7. 참고 문헌

[1] Cheonsu Jeong, "Generative AI service implementation using LLM application architecture: based on RAG model and LangChain framework," Journal of Intelligence and Information Systems, Dec. 2023.

[2] Github, langchain-kr,

[Online] Available: <https://github.com/teddylee777/langchain-kr>

[3] LangChain Documentation,

[Online] Available: https://python.langchain.com/docs/how_to

[4] You Tube, LangChain Tutorial,

[Online] Available: <https://www.youtube.com/watch?v=1scMJH93v0M>