



LLM

GitHub

Q&A

# GitHub 기반 코드 분석 시스템

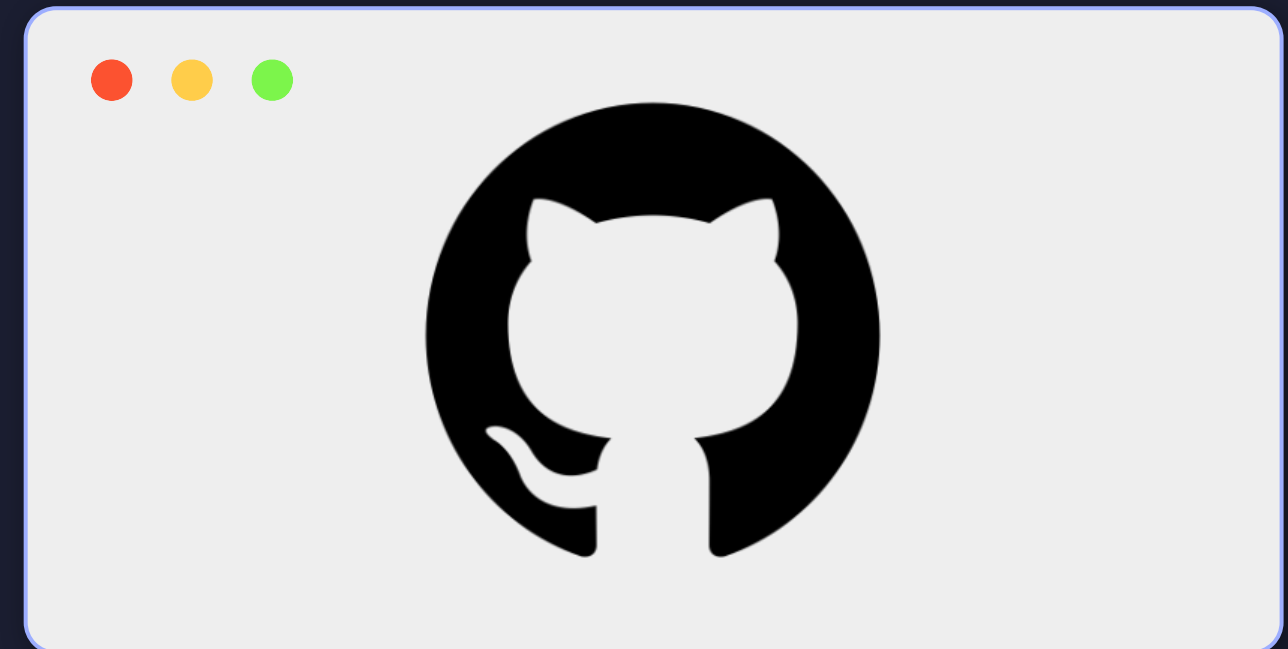
LLM을 연동한 내외부 문서 기반 질의 응답 시스템

SK Networks Family AI 12기

단위 프로젝트 3차

TEAM 3

김이경 노명구 지상원 허한결 황차해





LLM

GitHub

Q&A

Project Objective ; # 프로젝트 목적

Key Features ; # 챗봇 기능

Development Process ; # 챗봇 개발

Demonstration ; # 시연

Applications & Impact ; # 챗봇 활용 방안 및 프로젝트 의의



# 프로젝트 목적

nmmm9 / car\_registration

<> Code

Issues

Pull requests

Actions

Projects

Security

Insights

Files

main

Go to file

car\_registration\_system

app

pages

main.py

crawling

car\_data\_crawler.py

faq\_crawler.py

database

sql

db\_connector.py

models.py

.env.example

car\_registration / car\_registration\_system /

Add file

...

nmmm9 Initial commit

29a8c45 · 2 months ago

History

Name	Last commit message	Last commit date
..		
app	Initial commit	2 months ago
crawling	Initial commit	2 months ago
database	Initial commit	2 months ago
.env.example	Initial commit	2 months ago
car_registration_system.zip	Initial commit	2 months ago
requirements.txt	Initial commit	2 months ago
setup_db.py	Initial commit	2 months ago

팀원 중 한 명의 깃허브 주소를 복사하여 ChatGPT에 질문함  
해당 링크는 위와 같이 구성됨



# 프로젝트 목적

[https://github.com/nmmm9/car\\_registration/tree/main/car\\_registration\\_system](https://github.com/nmmm9/car_registration/tree/main/car_registration_system) 깃허브 주소야. 이 프로젝트의 목적은 무엇이며, 코드는 어떤 흐름으로 구성되어 있는지, 그리고 메인 함수는 어떤 파일에 있는지 알려줘

해당 GitHub 프로젝트는 차량 등록 시스템을 구현한 것으로 보입니다. 프로젝트의 목적은 차량 등록 프로세스를 디지털화하여 효율적으로 관리하는 것입니다. 이를 통해 차량 정보의 등록, 조회, 수정, 삭제 등의 기능을 제공하며, 사용자 친화적인 인터페이스를 통해 관리자의 업무를 간소화하고자 합니다.

코드의 구성은 일반적으로 다음과 같은 흐름을 따릅니다:

1. 데이터 모델 정의: 차량 정보에 대한 데이터 구조를 정의합니다.
2. 데이터베이스 연동: 정의된 모델을 기반으로 데이터베이스와 연동하여 정보를 저장하고 관리합니다.
3. 비즈니스 로직 구현: 차량 등록, 조회, 수정, 삭제 등의 핵심 기능을 구현합니다.
4. 사용자 인터페이스: 웹 또는 애플리케이션을 통해 사용자가 시스템과 상호작용할 수 있는 인터페이스를 제공합니다.
5. 메인 실행 파일: 전체 시스템을 실행하는 메인 함수가 포함된 파일로, 일반적으로 `main.py` 와 같은 이름을 사용합니다.

메인 함수는 보통 프로젝트의 루트 디렉토리나 `car_registration_system` 폴더 내의 `main.py` 파일에 위치합니다. 이 파일은 전체 애플리케이션의 진입점으로, 서버를 실행하거나 초기 설정을 담당합니다.

자세한 내용은 해당 GitHub 리포지토리의 `car_registration_system` 디렉토리에서 확인하실 수 있습니다:

📄 📁 🔍 🔄 ⌂ 🏠 출처

## ChatGPT 질의응답

Q. 프로젝트의 목적

A. 차량 정보의 등록, 조회, 수정, 삭제 등의 기능을 제공하는 등 차량 등록 프로세스의 효율적 관리를 위한다고 응답

-> 하지만, 실제로는 해당 내용의 프로젝트가 아님

Q. 코드의 흐름

-> 프로젝트의 실제 내용이 아닌, GPT가 추측한 내용으로 코드의 흐름을 답함

Q. 메인 함수의 위치

A. 자세한 내용은 해당 리포지토리의 디렉토리에서 확인

-> GPT에게 물어본 목적과는 다르게, 본인이 직접 확인해야 한다고 답함



## 프로젝트 목적

차량 등록, 조회, 수정, 삭제 등의 기능을 구현하는 코드는 어떤 이름의 파일에 있어?

GitHub 페이지를 불러오는 데 문제가 생겨 직접 확인은 어렵지만, 일반적으로 차량 등록, 조회, 수정, 삭제 기능은 다음과 같은 파일 이름에 포함되어 있을 가능성이 높습니다:

- `views.py`: 웹 애플리케이션의 비즈니스 로직이 담긴 파일로, 차량 관련 기능이 구현되어 있을 수 있습니다.
- `routes.py` 또는 `controllers.py`: 각 기능별 HTTP 요청 처리가 구현된 경우.
- `models.py`: 차량 데이터를 정의하고 데이터베이스와 연동하는 코드가 포함됩니다.
- `crud.py`: CRUD(Create, Read, Update, Delete) 기능이 분리되어 작성되는 경우.

해당 리포지토리를 정확히 분석하려면 `car_registration_system` 폴더 내부의 파일 목록을 직접 확인해보는 것이 좋습니다:

[GitHub 링크](#)

혹시 ZIP 파일로 받아 분석해드릴 수도 있어요. 원하시면 업로드해 주세요.



## ChatGPT 질의응답

Q. GPT가 주장하는 프로젝트 내용에 대해 코드의 위치를 물음

-> GitHub 페이지를 불러오는 데 문제가 생겨 직접 확인이 어렵고, 추측성의 답변을 함

-> 폴더 내부의 파일 목록을 직접 확인해보라고 응답



## 프로젝트 목적



car\_registration\_system.zip  
ZIP 아카이브

코드 분석해줘

압축을 풀어 확인한 결과, `venv` 가상환경을 제외하고는 Python 코드 파일(.py) 이 전혀 존재하지 않습니다. 즉:

- `car_registration_system.zip` 안에는 실행 가능한 실제 애플리케이션 코드가 포함되어 있지 않고,
- 대부분이 Python 가상환경(`venv`) 파일로 구성되어 있습니다.

### 가능한 상황

- 코드 파일이 압축에 포함되지 않았거나,
- 루트 폴더 위치가 잘못되어 코드가 다른 경로에 존재할 수도 있습니다.

### 다음 단계 제안

1. GitHub에서 다시 다운로드해 정확한 경로로 압축했는지 확인해 주세요.
2. 또는 `main.py` 나 `app.py`, `views.py` 등 핵심 코드 파일만 따로 업로드해 주시면 바로 분석해드릴 수 있습니다.

필요하시면 GitHub 리포지토리 주소로 다시 접속을 시도하거나, 로컬 경로를 알려주시면 구조를 맞춰드릴 수도 있어요. [↗]



## ChatGPT 질의응답

" 코드의 압축 파일을 제공하면 어떤 답변을 줄까? "

-> 파일 각각을 따로 업로드하면 분석해 줄 수 있다고 응답



## 프로젝트 목적

GPT는 GitHub가 어떤 구조로 구성되어있는지 확인하지 못함

찾는 기능이 어떤 파일에 포함되어 있는지 직접 확인해야 하며, GPT의 답변이 불분명하고 부정확

코드를 파악하기 위해서 각 파일을 각각 다운로드 후 GPT에 입력해야 하는 문제가 있음



GitHub Repository 링크와  
해당 Repository와 관련된 질문 (예. 리포지토리 구성, 코드 분석 등) 을 입력하면  
답변해주는 챗봇 개발



## 챗봇 기능

GitHub 내 프로젝트를 전반적으로 분석 예) 코드 분석, 원하는 함수 찾기 등

코드 수정 후 Git 커밋 및 푸시

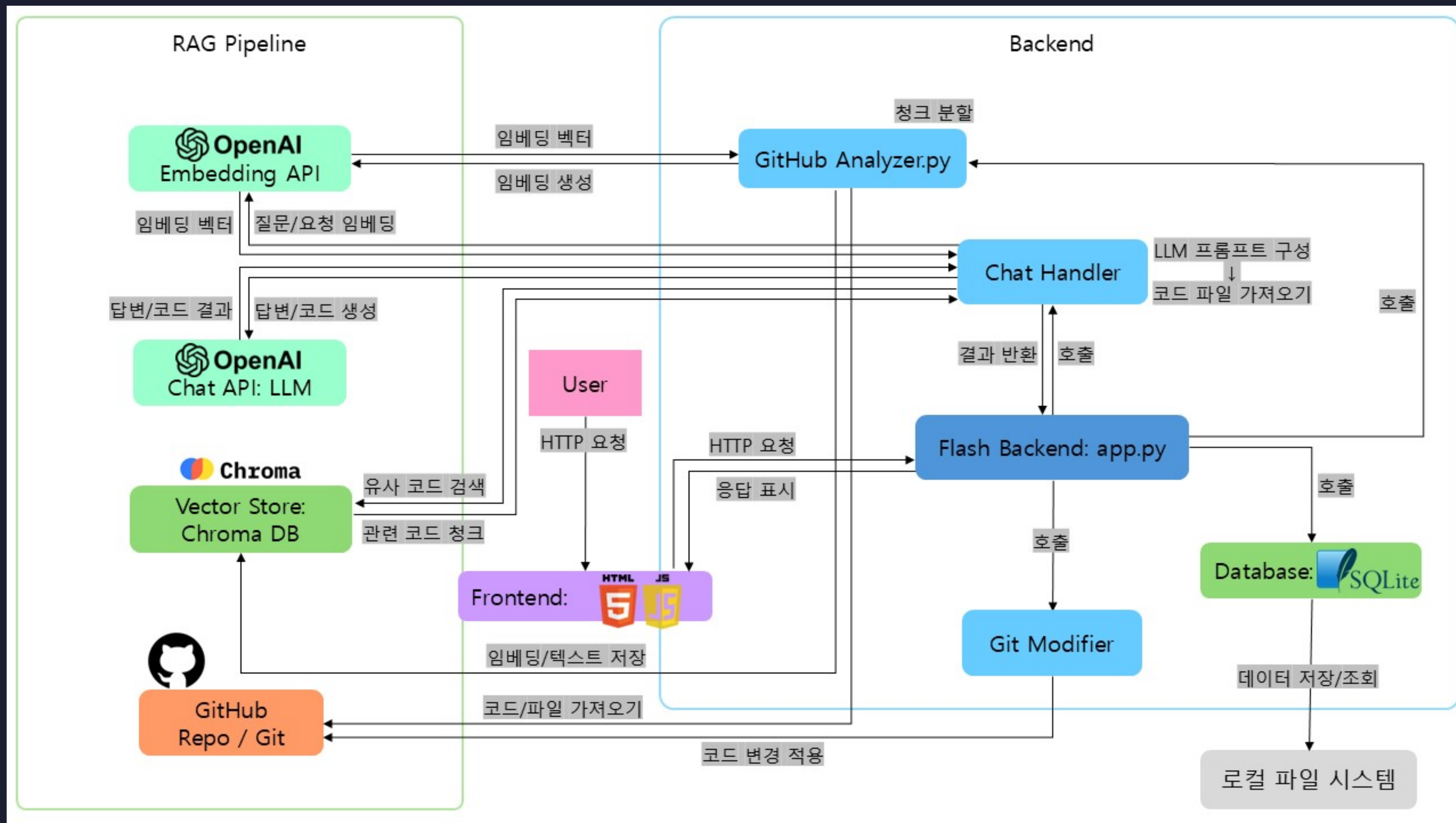
(4차에서 업그레이드 예정) 이전 대화기록 불러오기

(4차에서 업그레이드 예정) 회원 각자의 GitHub 접근





# 챗봇 개발





# 챗봇 개발

## 1단계 | GitHub 저장소 분석

- Flask라는 웹 서버가 `@app.route('/analyze')`를 통해 요청 받음
- gitpython 라이브러리를 이용해 GitHub 저장소를 로컬로 복제
- py, .js, .md 같은 분석 대상 파일만 선택
- 코드 파일을 작게 청크(chunk)로 나눔 → 검색이나 요약에 용이함
- Python은 함수/클래스 단위로, Markdown은 제목 단위로 쪼갬
- 청크마다 파일명, 함수명, 클래스명, 시작/끝 줄, 역할 태그 등 메타데이터를 붙임
- 각 청크에 대해 임베딩 (숫자 벡터로 변환)하고 ChromaDB에 저장



# 챗봇 개발

## 2단계 | 사용자 질문 처리

- Flask 라우트가 질문을 받아 → handle\_chat() 함수가 실행됨
- 질문 내용을 LLM에게 보내서 질문 의도(역할)를 파악  
예) “로그인 처리 기능을 찾고 싶다”
- 질문도 임베딩해서 관련된 청크를 찾아냄
- 1차: 역할 태그 기반 청크 검색
- 2차: 해당 파일 전체 로드
- 과거 대화 내용도 함께 분석해 맥락 유지



# 챗봇 개발

## 3단계 | 답변 생성

- LLM에게 보낼 프롬프트를 구성

  - 질문 내용

  - 관련 코드 청크들

  - 코드의 메타데이터 (파일명, 함수명, 줄 수, 역할 태그 등)

  - 디렉토리 구조

  - 이전 대화 기록

- 시스템 프롬프트: “당신은 GitHub 코드 분석 전문가입니다” ← 역할을 지정

- LLM이 질문에 대해 근거 있는 응답을 생성해서 사용자에게 반환함.



# 챗봇 개발

## 4단계 | 코드 수정 요청

- 어떤 코드 청크가 관련 있는지 찾을 (임베딩 + 역할 태그)
- LLM에게 “이 코드를 이렇게 바꿔줘”라고 지시
- LLM이 diff 형식(변경 전/후 줄 비교)으로 수정사항을 제안
- 사용자에게 미리보기 제공 → 확인하면 실제 파일에 반영
- Git에 커밋하거나, 원하면 GitHub에 푸시까지 가능

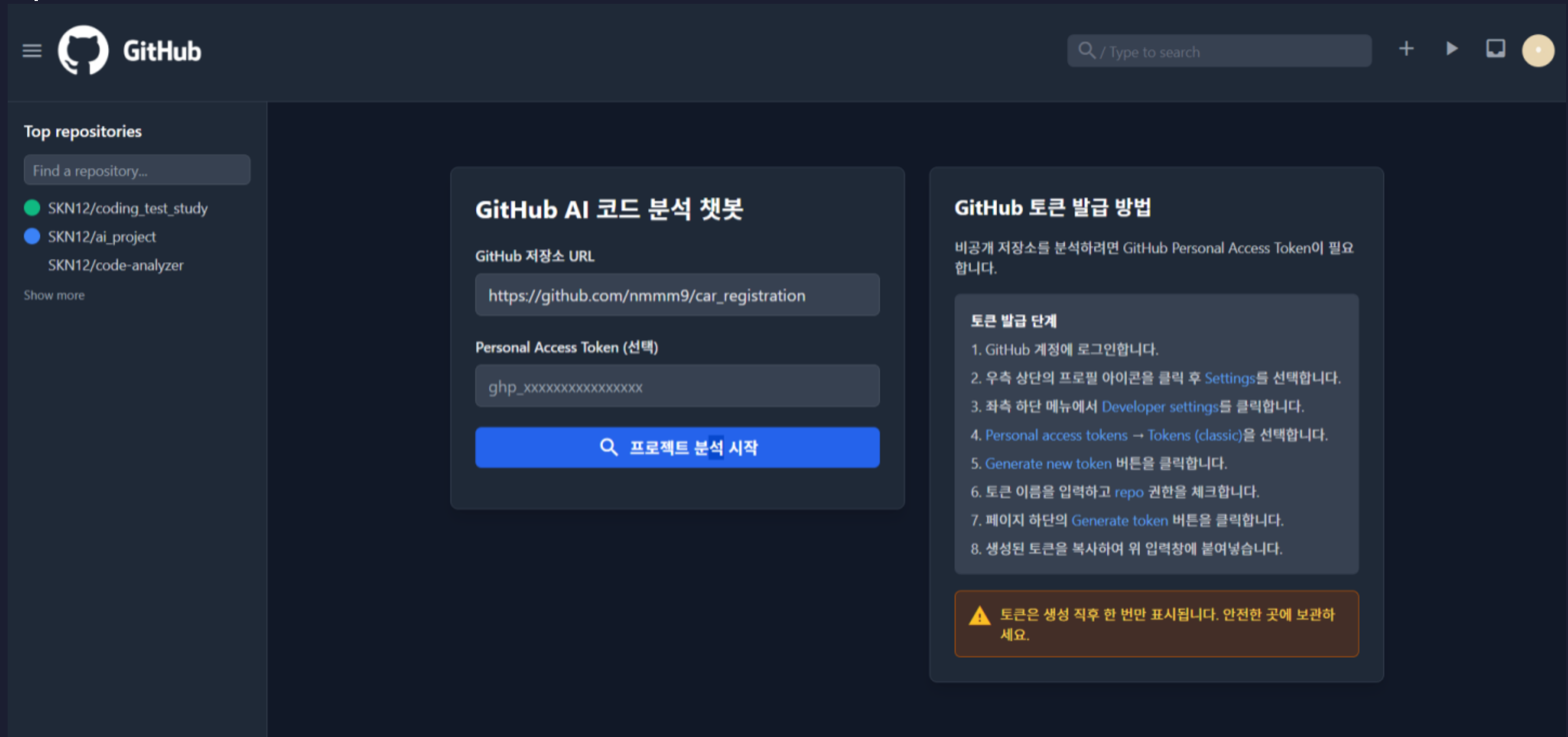


# 챗봇 개발

기술 요소	설명
Flask	웹 요청을 처리하는 Python 백엔드 프레임워크
GitPython	GitHub 저장소를 복제하고 관리
청크 분할 (Chunking)	코드를 함수, 클래스, 섹션 단위로 쪼갬
임베딩 (Embedding)	코드를 벡터로 변환해 의미 기반 검색 가능
역할 태그	코드나 질문의 의미를 "로그인", "UI", "데이터 처리" 등으로 분류
LLM 응답 생성	GPT에게 질문에 대한 답변 생성시키기
ChromaDB	임베딩된 코드 청크 저장용 벡터 데이터베이스
diff 적용 및 Git	코드 수정 후 Git으로 기록 및 푸시



# 시연 | 예시 1



이전에 ChatGPT에 질문했던 링크와 동일한 깃허브 리포지토리를  
현 프로젝트를 통해 개발한 챗봇에 질문



# 시연 | 예시 1

 GitHub

저장소 분석이 완료되었습니다. 질문해주세요!

안녕하세요! GitHub 저장소의 코드에 대해 질문하거나 코드 수정을 요청해보세요. 언제든지 도와드리겠습니다! 파일 구조, 함수의 목적, 버그 해결 방법 등 어떤 질문이든 가능합니다.

나: 이 프로젝트에 대해서 설명해줘

AI:

이 프로젝트는 자동차 등록 시스템으로, 다양한 기능을 제공하는 여러 모듈로 구성되어 있습니다. 프로젝트의 주요 기능과 구조는 다음과 같습니다:

프로젝트 구조 및 주요 기능

1. 디렉토리 구조

- app: 사용자 인터페이스와 관련된 페이지들이 포함되어 있습니다.
  - 01\_car\_registration.py: 자동차 등록과 관련된 페이지.
  - 02\_statistics.py: 제조사 및 모델 분석을 수행하는 페이지.
  - 03\_faq.py: FAQ 데이터를 시각화하고 새로운 FAQ를 추가하는 기능을 제공하는 페이지.
  - 03\_map\_view.py, 04\_used\_car\_analysis.py, 05\_rentcar\_analysis.py: 각각 지도 보기, 중고차 분석, 렌트카 분석과 관련된 페이지.
- crawling: 웹 크롤링 관련 모듈.
  - car\_data\_crawler.py: 자동차 데이터 크롤링.
  - faq\_crawler.py: 다양한 기업의 FAQ 페이지에서 질문과 답변을 스크래핑.
- database: 데이터베이스 관련 모듈.
  - sql: SQL 쿼리 및 테이블 생성 스크립트.
  - db\_connector.py: 데이터베이스 연결을 관리하는 모듈.
  - models.py: 데이터베이스 모델 정의.

2. 주요 기능 설명

- FAQ 관리 (03\_faq.py):
  - FAQ 시각화: 회사별 FAQ 수를 계산하고, 이를 그래프와 테이블 형식으로 시각화합니다.
  - 새로운 FAQ 추가: 사용자가 새로운 질문과 답변을 입력하여 FAQ 데이터베이스에 저장할 수 있습니다.
- 코드 예시:

# 회사별 질문 수 계산  
company\_stats = faq\_data.groupby('company\_name').size().reset\_index(name='FAQ 수')  
st.bar\_chart(company\_stats.set\_index('company\_name'))

- 제조사/모델 분석 (02\_statistics.py):

ChatGPT에서는 부정확한 답변을 얻은 반면,  
현 프로젝트로 개발한 챗봇에서는 레포지토리의 내용과 유사하게 답변하는 것을 확인





# 시연 | 예시 2



저장소 분석이 완료되었습니다. 질문해주세요!



안녕하세요! GitHub 저장소의 코드에 대해 질문하거나 코드 수정을 요청해보세요. 언제든지 도와드리겠습니다! 파일 구조, 함수의 목적, 버그 해결 방법 등 어떤 질문이든 가능합니다.

나: 이 프로젝트가 무슨 프로젝트인지 설명해줘

AI:  
이 프로젝트는 Llama라는 이름의 프로젝트로, 주로 자연어 처리(NLP)와 관련된 기능을 제공하는 것으로 보입니다. 프로젝트의 주요 목적과 기능은 다음과 같습니다:

1. 프로젝트 목적 및 기여 가이드라인:

- CONTRIBUTING.md 파일의 # Contributing to Llama 섹션에 따르면, 이 프로젝트는 기여를 쉽게 하고 투명하게 만들기 위한 가이드라인을 제공하고 있습니다. 이는 프로젝트가 오픈 소스 커뮤니티의 참여를 장려하고 있다는 것을 의미합니다.

2. 버그 및 이슈 관리:

- CONTRIBUTING.md의 ## Issues 섹션에서는 GitHub 이슈를 통해 공개적으로 버그를 추적하고 있으며, 버그를 재현할 수 있는 명확한 설명을 요구하고 있습니다. 이는 프로젝트가 버그 관리에 중점을 두고 있음을 나타냅니다.

3. 커뮤니티 및 행동 강령:

- CODE\_OF\_CONDUCT.md 파일의 ## Our Pledge와 ## Scope 섹션에서는 프로젝트와 커뮤니티 참여가 모든 사람에게 개방적이고 환영받는 환경이 되도록 노력하고 있음을 명시하고 있습니다. 이는 프로젝트가 커뮤니티의 다양성과 포용성을 중요시한다는 것을 보여줍니다.

4. 디렉토리 구조 및 파일 설명:

- llama 디렉토리에는 generation.py, model.py, tokenizer.py 등의 파일이 포함되어 있어, 이 프로젝트가 텍스트 생성, 모델 관리, 토큰라이저 기능을 제공하는 것을 알 수 있습니다.
- example\_chat\_completion.py와 example\_text\_completion.py 파일은 사용 예제를 제공하여, 사용자가 프로젝트의 기능을 쉽게 이해하고 활용할 수 있도록 돕습니다.

5. 기타 문서 및 정책:

- LICENSE, README.md, USE\_POLICY.md 등 다양한 문서 파일이 포함되어 있어, 프로젝트의 사용 정책, 라이선스, 업데이트 내역 등을 명확히 하고 있습니다.

이러한 정보들을 종합해 볼 때, Llama 프로젝트는 자연어 처리와 관련된 기능을 제공하며, 오픈 소스 커뮤니티의 참여를 장려하고, 다양한 문서와 정책을 통해 사용자의 이해를 돕고 있는 프로젝트로 이해할 수 있습니다.

해당 정보는 CONTRIBUTING.md, CODE\_OF\_CONDUCT.md 등의 파일에서 추출한 내용에 기반하고 있습니다. 추가적인 세부 사항은 프로젝트의 README.md 파일을 참조하면 더 많은 정보를 얻을 수 있을 것입니다.





















# 시연 | 예시 2

나: 프로젝트 디렉토리 구조에 대해서도 설명해줘

AI: 프로젝트 디렉토리 구조에 대해 설명드리겠습니다. 이 구조는 프로젝트의 파일과 폴더가 어떻게 조직되어 있는지를 보여주며, 각 파일과 폴더의 역할을 이해하는 데 도움이 됩니다.

프로젝트 디렉토리 구조

-  .github
  -  ISSUE\_TEMPLATE
    -  bug\_report.md: 버그 보고를 위한 템플릿 파일입니다. 사용자가 버그를 보고할 때 일관된 형식을 제공하여, 문제를 명확히 이해하고 해결하는 데 도움을 줍니다.
-  llama
  -  init.py: 이 파일은 llama 패키지를 초기화하는 역할을 합니다. 패키지를 임포트할 때 실행되는 기본 설정을 포함할 수 있습니다.
  -  generation.py: 텍스트 생성과 관련된 기능을 구현하는 파일로 추측됩니다. 구체적인 기능은 코드 내용을 확인해야 합니다.
  -  model.py: 모델의 구조와 동작을 정의하는 파일입니다. 모델의 학습, 예측 등의 기능을 포함할 수 있습니다.
  -  tokenizer.py: 텍스트를 토큰으로 변환하는 기능을 제공하는 파일입니다. 자연어 처리에서 중요한 역할을 합니다.
-  .gitignore: Git에서 추적하지 않을 파일이나 폴더를 지정하는 파일입니다. 예를 들어, 빌드 결과물이나 개인 설정 파일 등이 포함될 수 있습니다.
-  CODE\_OF\_CONDUCT.md: 프로젝트 참여자들이 따라야 할 행동 강령을 명시한 파일입니다. 프로젝트의 건강한 커뮤니티 유지를 위해 중요합니다.
-  CONTRIBUTING.md: 프로젝트에 기여하는 방법에 대한 가이드라인을 제공하는 파일입니다. 기여자가 프로젝트에 기여할 때 따라야 할 절차와 규칙을 설명합니다.
-  LICENSE: 프로젝트의 라이선스 정보를 포함하는 파일입니다. 사용자가 프로젝트를 어떻게 사용할 수 있는지를 명시합니다.
-  MODEL\_CARD.md: 모델의 특성과 사용법에 대한 정보를 제공하는 파일입니다. 모델의 성능, 제한 사항 등을 설명합니다.
-  README.md: 프로젝트에 대한 전반적인 설명을 제공하는 파일입니다. 프로젝트의 목적, 설치 방법, 사용법 등을 포함합니다.
-  Responsible-Use-Guide.pdf: 프로젝트의 책임 있는 사용을 위한 가이드를 제공하는 PDF 문서입니다.
-  UPDATES.md: 프로젝트의 업데이트 내역을 기록하는 파일입니다. 변경 사항, 버그 수정 등을 포함할 수 있습니다.
-  USE\_POLICY.md: 프로젝트의 사용 정책을 명시한 파일입니다. 사용자가 프로젝트를 사용할 때 따라야 할 규칙을 설명합니다.
-  download.sh: 프로젝트와 관련된 파일이나 데이터를 다운로드하는 스크립트 파일입니다.



## 챗봇 활용 방안 및 프로젝트 의의

- GitHub 초보자 및 비전공자의 학습에 도움
  - AI 대화형 분석으로 개발 경험 없는 사용자도 빠르게 프로젝트 구조 파악 가능
- 문서 부족, 기능 이해 어려움 문제 해결 등 실질적인 협업에 도움
  - GPT를 코드 리뷰어 + 리팩토링 도우미 + 설명자로 활용
- 의미 기반 검색 시스템 구현
  - 질문과 코드에 역할 태그 자동 부여 → 단순 키워드 검색보다 정확한 맥락 기반 응답 제공
- 코드 어시스턴트
  - LLM 수정 제안 → 사용자 확인 → Git 커밋 자동 처리
  - 실질적인 AI 기반 프로그래밍 도우미 실현